I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° 1918

# THE PRICE OF ANONYMITY: OPTIMAL CONSENSUS DESPITE ASYNCHRONY, CRASH AND ANONYMITY

FRANÇOIS BONNET  MICHEL RAYNAL

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

# The Price of Anonymity: Optimal Consensus despite Asynchrony, Crash and Anonymity

François Bonnet* Michel Raynal**

Systèmes communicants
Projet ASAP

Publication interne n ˚ 1918 — Décembre 2008 — 30 pages

**Abstract:** This paper addresses the consensus problem in asynchronous systems prone to process crashes, where additionally the processes are anonymous (they cannot be distinguished one from the other: they have no name and execute the same code). To circumvent the three computational adversaries (asynchrony, failures and anonymity) each process is provided with a failure detector of a class denoted $\psi$, that gives it an upper bound on the number of processes that are currently alive (in a non-anonymous system, the classes $\psi$ and $\mathcal{P}$ -the class of perfect failure detectors- are equivalent).

The paper first presents a simple $\psi$-based consensus algorithm where the processes decide in $2t+1$ asynchronous rounds (where $t$ is an upper bound on the number of faulty processes). It then shows one of its main results, namely, $2t+1$ is a lower bound for consensus in the anonymous systems equipped with $\psi$. The second contribution addresses early-decision. The paper presents and proves correct an early-deciding algorithm where the processes decide in $\min(2f+2, 2t+1)$ asynchronous rounds (where $f$ is the actual number of process failures). This leads to think that anonymity doubles the cost (wrt synchronous systems) and it is conjectured that $\min(2f+2, 2t+1)$ is the corresponding lower bound.

The paper finally considers the $k$-set agreement problem in anonymous systems. It first shows that the previous $\psi$-based consensus algorithm solves the $k$-set agreement problem in $R_t = 2\lfloor \frac{t}{k} \rfloor + 1$ asynchronous rounds. Then, considering a family of failure detector classes $\{\psi_\ell\}_{1 \leq \ell \leq k}$ that generalizes the class $\psi(= \psi_1)$, the paper presents an algorithm that solves the $k$-set agreement in $R_{t,\ell} = 2\lfloor \frac{t}{k-\ell+1} \rfloor + 1$ asynchronous rounds. This last formula relates the cost ($R_{t,\ell}$), the coordination degree of the problem ($k$), the maximum number of failures ($t$) and the the strength ($\ell$) of the underlying failure detector. Finally the paper concludes by presenting problems that remain open.

**Key-words:** Anonymity, Asynchronous system, Consensus, Distributed computability, Early decision, Failure detector class, Message passing system, Round-based computation, Set agreement.

*(Résumé : tsvp)*

* IRISA, Université de Rennes 1, Campus de Beaulieu, 35042 Rennes Cedex, France francois.bonnet@irisa.fr
** IRISA, Université de Rennes 1, Campus de Beaulieu, 35042 Rennes Cedex, France, raynal@irisa.fr

Centre National de la Recherche Scientifique (UMR 6074) Université de Rennes 1 – Insa de Rennes
Institut National de Recherche en Informatique et en Automatique – unité de recherche de Rennes

# Accord en dépit de l’anonymat

**Résumé :** Ce rapport étudie les problèmes d’accord dans les systèmes asynchrones anonymes sujets au crash des processus.

**Mots clés :** Accord, Anonymat, Asynchronisme, Défaillances.

# 1 Introduction

**Anonymous systems** While (in a somewhat restrictive way) the aim of a real-time system is to master *on time computing*, and the main aim of parallelism is to obtain *efficient* algorithms, we can say that the central issue of distributed computing consists in *mastering uncertainty*. This uncertainty has first appeared under the form of asynchrony, failure occurrences, and the multiplicity of loci of control (also referred as locality). More recently, new facets of uncertainty (such as dynamicity, scalability and mobility) have appeared and made distributed computing even more challenging.

Among the many facets of uncertainty that distributed computing has to cope with, *anonymity* is particularly important. It occurs when the computing entities (processes, agents, sensors, etc.) have no name, and consequently cannot distinguish the ones from the others. It is worth noticing that, from a practical point of view, anonymity is a first class property as soon as one is interested in guaranteeing privacy. As an example, some peer-to-peer file-sharing systems assume the peers are anonymous [15]. In the same vein, not all the sensor networks assume that each sensor as a proper identity [3, 18].

One of the very first works (to our knowledge) that addressed anonymous systems is from D. Angluin [2]. In that paper, considering message passing systems, she was mainly interested in computability issues, namely answering the question "which functions can be computed in presence of asynchrony and anonymity?" The leader election problem is a simple example of a problem that is unsolvable in such a setting (intuitively, this because symmetry cannot be broken in presence of asynchrony and anonymity). Other works have then addressed anonymity in particular settings such as ring networks [7], or networks with a regular structure [32]. Failure-free message passing anonymous systems have also been investigated in [43, 44] where is given a characterization of problems solvable in this context according to which amount on information about network attributes are initially known by the processes. Relations between broadcast and shared memory in reliable anonymous distributed systems are addressed in [4].

Failure-free asynchronous shared memory systems have been studied in the context of anonymity. A characterization of the problems (tasks) that can be solved in this setting (when additionally the number of processes is not known) is given in [6]. The use of randomization to cope with crash-prone anonymous shared memory systems has been addressed in [40], where a randomized wait-free naming algorithm is given that solves the naming problem when each atomic register is a single-writer/multi-reader register. Recently, wait-free algorithms implementing snapshot and weak counters have been proposed for anonymous asynchronous shared memory systems prone to process crash [25]. *Wait-free* means that every non-faulty process has to terminate its snapshot or counter operations, whatever the number of failures and and the concurrency pattern [28].

**Consensus in anonymous shared memory systems** Consensus is one of the most famous distributed computing problem. It is a coordination problem defined as follows: each process proposes a value, and each non-faulty process has to decide a value (termination), such that no two processes decide different values (agreement) and the decided value is a proposed value (validity). While it has a very simple statement and can be trivially solved in (anonymous or not) failure-free systems where the number of processes is known, and has simple solutions in (anonymous or not) crash-prone synchronous systems, the consensus problem has no solution in asynchronous non-anonymous failure-prone systems, as soon as (even only) one process can be faulty, be the failure a simple crash and the communication system a reliable shared memory system [33], or a reliable message passing system [22]. Trivially, the problem cannot be solved either if anonymity is added to asynchrony and failures.

An approach based on randomization is presented in [10] to circumvent the previous impossibility in anonymous crash-prone shared memory systems. As noticed in [25], this shows that producing unique iden-

tifiers is harder than consensus. Interestingly, this approach has been extended to infinitely many processes in [5].

Another approach to circumvent the previous impossibility consists in considering a weaker version of the problem. Taking such an approach, [25] looks for obstruction-free consensus algorithms. *Obstruction-freedom* is a termination property weaker than wait-freedom. While (in the consensus context) wait-freedom requires that every non-faulty process always decides (see above), obstruction-freedom [29] requires that, whatever the failure pattern, each non-faulty process $p$ decides when the concurrency pattern is such that $p$ can execute "long enough" without concurrency. (From a practical point of view, "long enough" means the time to execute its algorithm.) An obstruction-free consensus algorithm for anonymous shared memory systems is described in [25]. This algorithm requires $O(n)$ binary atomic registers (where $n$ is the total number of processes).

**Content of the paper** As far as we know, the consensus problem has not been investigated in anonymous crash-prone *message passing* systems. This is the topic addressed in this paper. Several contributions are presented. The first is a failure detector-based algorithm that solves the consensus problem despite the net effect of asynchrony, crash failures and anonymity. The second (and, to our view, a main contribution) is a lower bound on the number of rounds required by any algorithm that solves consensus in such an uncertainty context. The third is an early-deciding algorithm while the last contribution is the investigation of the $k$-set agreement problem in anonymous systems.

As consensus cannot be solved in presence of process crashes and asynchrony in a message passing systems [22], these systems have to be enriched with additional power in order the problem becomes solvable. Failure detectors are a well-known approach proposed to provide processes with such an additional power [11]. Informally, a failure detector provides each process with information on failures. As we are interested in the most efficient asynchronous message passing algorithm that solves consensus despite crashes and anonymity (and not in the weakest $\Omega$-like [12] failure detector to face anonymity), we consider here the failure detector class denoted $\psi$. That failure detector class is the strongest of a family of failure detector classes that has been introduced in [37]. When queried by a process, such a failure detector returns an over-estimate of the number of alive processes. (A simple combination of results established in [37] and [38] shows that $\psi$ and the class $\mathcal{P}$ of perfect failure detectors are equivalent in asynchronous non-anonymous systems. A failure detector of the class $\mathcal{P}$ provides each process with a set that does not contain the id of a process before it crashes and eventually contains the ids of all the crashed processes.) Interestingly, both $\psi$ and $\mathcal{P}$ are classes of realistic failure detectors. (A realistic failure is a failure detector that can be implemented in a synchronous system. Said differently, a realistic failure detector cannot guess the future [16].)

The paper first presents an asynchronous anonymous $\psi$-based algorithm that solves the consensus problem in $2t + 1$ rounds, where $t$ is an upper bound on the number of processes that are allowed to crash in a run ($1 \leq t \leq n - 1$). Its design principle is pretty simple, namely the algorithm applies the well-known flood-set strategy to the anonymous context. Then the paper presents one of its results, namely, a proof that, whatever the crash failure pattern, $2t + 1$ is a lower bound on the number of rounds required to solve the consensus problem in the proposed round-based model.

While $t + 1$ is a lower bound on the number of rounds to solve consensus in both synchronous message passing systems [1, 21, 31], and asynchronous message passing systems equipped with a perfect failure detector [27], it appears that $2t + 1$ is the corresponding lower bound for anonymous systems. This is a noteworthy feature of anonymity as it shows that, when one wants to solve consensus despite anonymity, an additional price of $t$ rounds has to be paid The lesson learned is that, from a time complexity point of view, the combination of asynchrony and anonymity doubles the price. (Let us notice that, in a synchronous system, consensus can easily be solved with a classical flood-set algorithm in $(t + 1)$ rounds).

The paper then considers early decision in anonymous systems enriched with $\psi$. It presents an algorithm where the processes decide and halt by $\min(2f+2, 2t+1)$ rounds (where $f$ is the actual number of faulty processes, $0 \leq f \leq t$). This leads to think that $\min(2f+2, 2t+1)$ could be the lower bound on the number of rounds for solving consensus in these asynchronous systems.

Finally, the paper focuses on the $k$-set agreement problem [13] that extends consensus in the sense it allows up to $k$ values to be decided. It first shows that the previous $\psi$-based algorithm (designed for consensus) solves $k$-set agreement in $2\left\lfloor \frac{t}{k} \right\rfloor + 1$ rounds. As $k$-set agreement is a weaker problem than consensus, a failure detector weaker than $\psi$ should be able to solve it. To investigate this idea, a family of failure detector classes, denoted $\{\psi_\ell\}_{1 \leq \ell \leq n}$, is introduced; $\psi_1$ is $\psi$, and $\psi_{\ell+1}$ is weaker than $\psi_\ell$. It is shown that $\ell \leq k$ is a sufficient requirement to solve $k$-set agreement with the help of $\psi_\ell$. Moreover a $\psi_\ell$-based $k$-set agreement is presented that requires $R = 2\lfloor \frac{t}{k-\ell+1} \rfloor + 1$ rounds. Interestingly, this formula relates the cost ($R$), the coordination degree of the problem ($k$), and the strength ($\ell$) of the underlying failure detector. It also clearly exhibits the point until which the failure detector class can be weakened while still solving $k$-set agreement, namely $\ell$ cannot be greater than $k$ (the threshold value $\ell = k+1$, i.e. the value from which $\psi_\ell$ is too weak to solve $k$-set agreement, corresponds to a division by 0 in the formula).

**Roadmap** The paper is made up of 7 sections. Section 2 presents the system model which includes the failure detector class $\psi$. Section 3 presents and proves correct a simple $\psi$-based algorithm that solves consensus despite asynchrony, process crashes and anonymity. Then, Section 4 proves a main theorem of the paper: $(2t+1)$ is a lower bound on the number of rounds for any algorithm that solves the consensus problem in that computation model. Then, Section 5 addresses early decision, and Section 6 focuses on the $k$-set agreement problem. Finally, Section 7 concludes the paper.

# 2 Computation model

## 2.1 Base model

**Process model** The system is made up of a fixed number $n$ of processes, denoted $p_1, \ldots, p_n$. The value of the system parameter $n$ is not known by the processes. Moreover, the process $p_i$ does not know its index $i$, which means that indexes are only used for a presentation purpose. Processes are anonymous in the sense that they have no name and execute the same algorithm. They are asynchronous in the sense that there is no assumption on their respective speeds.

**Failure model** Up to $t$ processes can crash in a run, $0 \leq t \leq n-1$. A process executes correctly its algorithm until it possibly crashes. A crash is a premature stop; after it has crashed, a process executes no statement. The value of the system parameter $t$ is know by the processes. A process that does not crash in a run is *correct* in that run. Otherwise, it is ***faulty*** in that run.

**Communication** The processes communicate by exchanging messages through reliable channels. These channels are asynchronous, which means that there is no assumption on the speed of messages on channels, except that it is positive (eventually every message arrives).

The processes are provided with a brodcast() communication primitive that allows the invoking process to send the same message to all the processes (including itself). The brodcast() primitive is not reliable in the sense that, if a process $p_i$ crashes while broadcasting a message, that message can be received by an arbitrary subset of processes. When it receives a message, a process cannot determine the sender of the message. Moreover, given any set of messages it has received, a process cannot determine if these messages are from the same sender or from different senders.

**Round-based model** The processes execute asynchronous rounds. During each round, a process broadcasts a message, receives messages sent during the very same round and executes local computation. This means that, as in the asynchronous models described in [8, 23, 34], the rounds are communication-closed [20].

**Notation** The previous computation model is denoted $\mathcal{AARS}^{cl}_{n,t}[\emptyset]$. $\mathcal{AARS}$ stands for $\mathcal{A}$nonymous $\mathcal{A}$synchronous $\mathcal{R}$ound-based $\mathcal{S}$ystem, with communication-$cl$osed rounds, while $\emptyset$ means there is no additional assumption.

## 2.2 The failure detector class $\psi$

As indicated in the introduction this failure detector class has been introduced in [37]. The class $\psi$ is the equivalent of the class of perfect failure detectors $\mathcal{P}$, when we consider non-anonymous systems ("equivalent" means that, if we associate distinct names with each process of an anonymous system, we have the following: given a failure detector of any one class it is possible to build a failure detector of the other class [37, 38]).

**Definition** Let $f$ denote the number of processes that crash in a given run ($0 \leq f \leq t$), and $f^\tau$ denote the number of processes that have crashed up to time $\tau$. A failure detector of the class $\psi$ provides each process $p_i$ with a positive integer denoted $aa\ell_i$ ($a$pproximate number of $a\ell$ive processes) that satisfies the following properties (where $aa\ell_i^\tau$ denotes the value of $aa\ell_i$ at time $\tau$):

- Safety: $\forall \tau :\ aa\ell_i^\tau \geq n - f^\tau$.
- Liveness: $\exists \tau :\ \forall \tau' \geq \tau :\ aa\ell_i^{\tau'} = n - f$.

The safety property states that $aa\ell_i$ is always an over-estimate of the number of processes that are still alive, while the liveness property states that it eventually converges to its exact value[1].

## 2.3 The computation model $\mathcal{AARS}^{cl}_{n,t}[\psi]$

This computation model is $\mathcal{AARS}^{cl}_{n,t}[\emptyset]$ enriched with $\psi$ and where, in each round, the number of messages received by a process $p_i$ is determined by the current value of $aa\ell_i$. More precisely, for each process $p_i$, the algorithms have the canonical form described at the right. The local variable $r_i$ is the current round number of $p_i$. Each process $p_i$ execute asynchronous rounds until some condition is satisfied. During its round $r_i$, $p_i$ broadcast a message tagged $r_i$, waits until it has received $aa\ell_i$ messages tagged $r_i$, and executes local computation. ($aa\ell_i$ is repeatedly read until the wait statement terminates.) Before proceeding to the next round, the process $p_i$ increases $r_i$. (As the model is asynchronous it is up to each process $p_i$ to manage its round number).

$r_i \leftarrow 1$;
**while** ($\neg$ condition) **do**
  **begin asynchronous round**
  broadcast a msg tagged $(r_i, -)$;
  **wait until** ($aa\ell_i$ msgs tagged $r_i$ have been received);
  Local computation;
  $r_i \leftarrow r_i + 1$
  **end asynchronous round**
**end while**;
Local computation.

**Misleading notification** Let us consider Figure 1 where the rounds $r-1$, $r$ and $r+1$ are represented, the processes $p_a$ crashes during the round $r-1$ (a crash is represented by a cross in the figure), and the process $p_b$ crashes after it has broadcast its round $r$ message (in the figure, the corresponding crash appears during the round $r+1$). The asynchronous notification of each crash appears at $p_i$ as a decrease of $aa\ell_i$; each is indicated with a dotted line. As $p_a$ crashes during the round $r-1$, it will not send round $r$ messages, and so, during the round $r$, $p_i$ has to wait for at least 3 messages ($aa\ell_i = 3$). Differently, $p_i$ is notified of the crash

[1] In [37], $n$ is known and $\psi$ provides each process $p_i$ with an integer $anc_i$ such that $n = aa\ell_i + anc_i$.

of $p_b$ (i.e., $aa\ell_i$ is decreased to 2) while it is waiting for round $r$ messages. As a result $p_i$ waits for only two messages and, as it has received two round $r$ messages (from $p_b$ and itself), it terminates its participation to the round $r$. Such an early failure notification is called a *misleading* notification, and the message $m$ sent by the corresponding crashed process is called a misleading message. More precisely, a message $m$ sent at round $r$ is *misleading* if it allows its receiver to terminate its round $r$, while the corresponding sender has crashed after or during the broadcast of $m$. These misleading notifications/messages come from the independence between the asynchronous communication-closed rounds on one side, and the crash notifications supplied by failure detector $\psi$ on the other side (it is such an independence that makes the system different from a synchronous system).

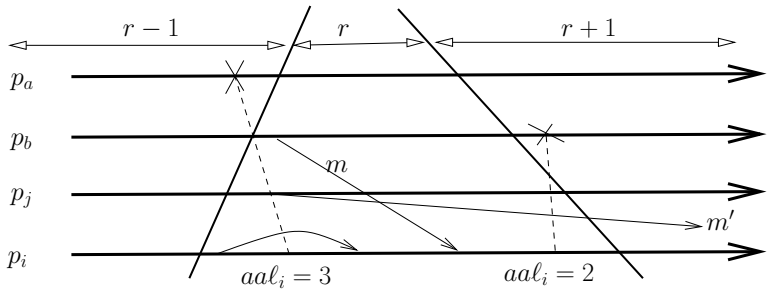



Figure 1: Misleading notification

The following theorem captures the synchronization power of $\psi$ in this round-based model.

**Theorem 1** *If $x$ processes crash while they execute the round $r$, no process can proceed to the round $r+1$ while there are still $(x+1)$ processes that are alive and execute the round $r-1$.*

**Proof** let $\tau$ be the time at which the first process (say $p_i$) progresses from the round $r$ to the round $r+1$. Moreover, let $A(\tau)$ be the number of processes that are alive at time $\tau$, and $R(\tau, r)$ be the number of processes that, at time $\tau$, have entered a round $r' \geq r$. We have $R(\tau, r) = RA(\tau, r) + RC(\tau, r)$ where $RA(\tau, r)$ is the number of processes that, at $\tau$, are alive and execute a round $r' \geq r$ (notice that only $p_i$ starts executing $r' = r+1$, the other processes of $RA(\tau, r)$ are executing $r$), and $RC(\tau, r)$ is the number of processes that have started executing the round $r$ and have crashed by time $\tau$).

- It follows from the safety property of $\psi$ that, when the process $p_i$ progresses from the round $r$ to the round $r+1$, we have $aa\ell_i(\tau) \geq A(\tau)$. Moreover, during the round $r$, $p_i$ receives and processes only messages sent during the same round $r$, from which we conclude that $R(\tau, r) \geq aa\ell_i(\tau)$, and by transitivity we obtain $R(\tau, r) \geq A(\tau)$.
- At time $\tau$, there are $A(\tau) - RA(\tau, r)$ alive processes that have not yet entered the round $r$. As $RA(\tau, r) = R(\tau, r) - RC(\tau, r)$ and $0 \leq RC(\tau, r) \leq x$, we conclude that there are at most $A(\tau) - R(\tau, r) + x$ alive processes that have not yet entered the round $r$.

Finally, as, at time $\tau$, there are at most $A(\tau) - R(\tau, r) + x$ alive processes that have not yet entered the round $r$, and $R(\tau, r) \geq A(\tau)$, we conclude that $A(\tau) - R(\tau, r) + x \leq x$, which completes the proof of the theorem. $\square_{Theorem\ 1}$

The corollary that follows considers the case $x = 0$.

**Definition 1** *We say that a process $p_i$* terminates a round $r$, *if $r < 2t+1$ and $p_i$ proceeds to $r+1$, or $r = 2t+1$ and $p_i$ decides during that round.*

**Corollary 1** *If no process crashes while executing round $r$, no process terminates the round $r$ while there are alive processes executing the round $r-1$.*

# 3 Solving consensus in $\mathcal{AARS}^{cl}_{n,t}[\psi]$

## 3.1 The algorithm

A consensus algorithm for the $\mathcal{AARS}^{cl}_{n,t}[\psi]$ model is described in Figure 2. This algorithm is a simple enrichment of the skeleton described in the previous section that adapts to $\mathcal{AARS}^{cl}_{n,t}[\psi]$ the classical flood set consensus algorithm designed for synchronous system [8, 34, 41].

A process $p_i$ invokes propose$(v_i)$ where $v_i$ is the value it proposes to the consensus. It terminates when it executes the return$(est_i)$ statement (line 10) where $est_i$ is the value it decides. The processes execute $(2t+1)$ asynchronous rounds (line 02). In each round, each process $p_i$ broadcasts its current estimate (denoted $est_i$ and initialized to $v_i$) of the decision value and updates it (by taking the minimum on the values it has received and taken into account up to now, lines 05-06).

```
operation propose(v_i):
(01) est_i ← v_i; r_i ← 1;
(02) while (r_i ≤ 2t + 1) do
(03)    begin asynchronous round
(04)    brodcast EST(r_i, est_i);
(05)    wait until ( aal_i messages EST(r_i, −) have been received );
(06)    est_i ← min(est values received at the previous line);
(07)    r_i ← r_i + 1;
(08)    end asynchronous round
(09) end while;
(10) return(est_i).
```

Figure 2: Anonymous consensus in $\mathcal{AARS}^{cl}_{n,t}[\psi]$

**Remark** If $n$ is known by the processes, the algorithm can be improved to reduce the number of rounds in the particular case where $t = n - 1$ (wait-free case). Instead of $(2t + 1)$ rounds, the processes can then execute only $2t$ rounds.

## 3.2 Proof of the algorithm

**Lemma 1** *A decided value is a proposed value (validity).*

**Proof** The proof of the validity property is a direct consequence of the following simple observations: (1) each local estimate $est_i$ is initialized to a proposed value (line 01), (2) only estimate values are communicated (lines 04-05), and (3) the new value of an estimate local variable is the minimum of the estimates values received and taken into account so far (lines 06). $\square_{Lemma\ 1}$

**Lemma 2** *Every correct process decides in* $(2t + 1)$ *rounds (termination).*

**Proof** Let us first observe that, due to the liveness properties of $\psi$, during any round $r$ no process can be blocked forever at line 05. The lemma follows directly from this observation: every process that does not crash by the end of the $(2t + 1)$th round decide at line 10. $\square_{Lemma\ 2}$

**Lemma 3** *If no process crashes during two consecutive rounds* $r$ *and* $r + 1$*, then all the processes that terminate the round* $r$ *have the same estimate value.*

**Proof** Let $r$ and $r+1$ be two consecutive rounds without crash, and $AR$ the set of processes that execute these two rounds. Let first observe that, due to Corollary 1 no process is alive in the round $r-1$ when a process of $AR$ proceeds to $r+1$, and as no process crashes while executing the round $r+1$, there is no misleading message. Finally, due to the safety property of $\psi$ and the fact that no process that crashed before $r$ can send round $r$ messages, it follows that each process in $AR$ receives a round $r$ message from each process in $AR$ and does not receive a round $r$ message from any process not in $AR$. Consequently, during the round $r$, the processes of $AR$ compute their new estimate as the smallest value from the same set, which proves the lemma. $\square_{Lemma\ 3}$

**Lemma 4** *No two processes decide different values (agreement).*

**Proof** We consider two cases.

- Case 1. In the sequence of $(2t+1)$ rounds, there are two consecutive rounds without crash. Let $r$ and $r+1$ be these two rounds, with $\leq r \leq 2t$. It follows from Lemma 3 that all the processes that proceed to the round $r+1$ have the same estimate value. Hence, a single value can be decided at the end of the round $(2t+1)$.
- Case 2. In the sequence of $(2t+1)$ rounds, there are no two consecutive rounds without crash. This means that the odd rounds are crash-free, while each even round has exactly one crash. So, the $t$ possible crashes occurred during the rounds $2, 4, \ldots, 2t$. As the last round is crash-free and there are no misleading messages during the round $2t+1$, it follows (from the safety property of $\psi$) that during that round every process receives a message from every process. They all consequently compute the same minimum value, which completes the proof of the lemma.

$\square_{Lemma\ 4}$

**Theorem 2** *The algorithm described in Figure 2 solves the consensus problem in* $(2t+1)$ *rounds in the* $\mathcal{AARS}^{cl}_{n,t}[\psi]$ *model.*

**Proof** The proof follows from the lemmas 1, 2 and 4. $\square_{Theorem\ 2}$

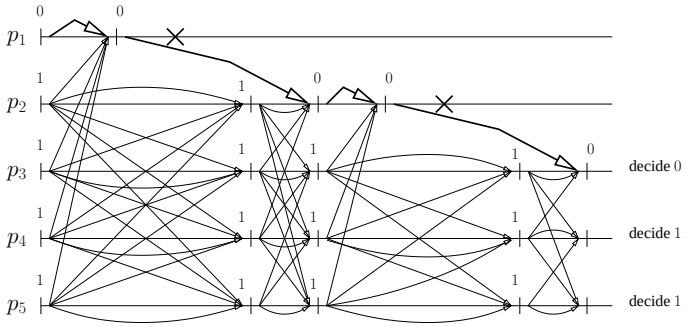



Figure 3: The algorithm requires $(2t+1)$ rounds

### 3.3 Why the algorithm requires $(2t+1)$ rounds

A simple run of the previous algorithm is depicted in Figure 3. This run, that considers $t=2$, involves $n=5$ processes and requires $2t+1=5$ rounds for the non-faulty processes to decide. In that run, the process denoted $p_1$ crashes during the second round, and the process denoted $p_2$ crashes during the fourth

round. Moreover, the message sent by $p_1$ during the first round is received only by itself, and its round 2 message is received only by $p_2$. Similarly for $p_2$ during the rounds 3 and 4. As we can see, if the algorithm stops at the 4th round, it does not solves the consensus problem.

The behavior of the chain of processes $p_1, p_2$ during the first four rounds (where the bold arrows are associated with the messages that carry the value 0) is similar to the chain of faulty processes that delays the decision until the round $(t+1)$ in synchronous systems. The next section shows that the $(2t+1)$ rounds price is not a particular feature of the proposed algorithm, but a feature of all the algorithms that solve the consensus problem in $\mathcal{AARS}^{cl}_{n,t}[\psi]$.

## 4 $(2t+1)$ is a lower bound

Assuming $t < n-1$, this section shows that $(2t+1)$ is a lower bound on the number of rounds to solve the consensus problem in both the model $\mathcal{AARS}^{cl}_{n,t}[\psi]$ and the model $\mathcal{AARS}^{op}_{n,t}[\psi]$ described in the next section devoted to early decision (the main difference is that $\mathcal{AARS}^{op}_{n,t}[\psi]$ is not round communication-closed).

The proof is by contradiction. Assuming that there is an algorithm $A$ that solves the binary consensus problem in $2t$ rounds, it shows that such an algorithm cannot be designed. (In the binary consensus problem, only the values 0 and 1 can be proposed by the processes. It is easy to see that considering only binary consensus can be done without loss of generality.)

**Definitions** A *configuration* is a global state of the considered consensus algorithm. The notion of *valence* has been introduced and used for the first time in [22]. A configuration $C$ is 0-*valent* (resp., 1-*valent*) if, starting from $C$, the only value that can be decided is 0 (resp., 1). A *univalent* configuration is either 0-valent or 1-valent. A *bivalent* configuration is a configuration that is not univalent (this means that, starting from a bivalent configuration there is a run of the algorithm that decides 0, and another run that decides 1).

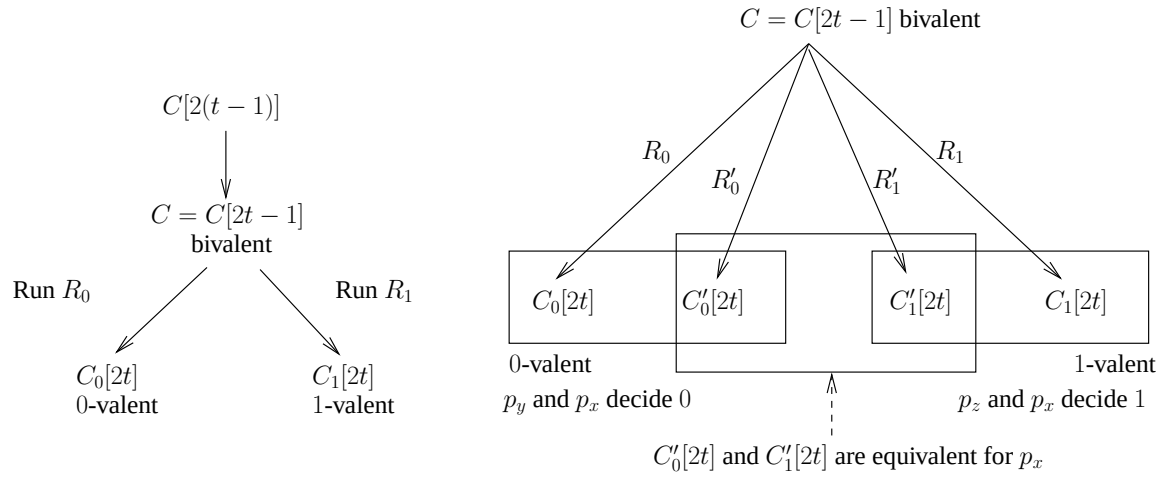
Here we are interested in the configurations $C[r]$ (where $r \geq 0$ is a round number) defined as follows. Let $\tau_r$ be the first time at which there is no more process alive in a round $\leq r$, i.e., $\tau_r$ occurs when the last alive process in a round $\leq r$ either crashes or proceeds to the round $r+1$ (in that case, the process was in the round $r$). $C[r]$ is the state of the considered algorithm at time $\tau_r$. Let us notice that, in $C[r]$, $r \geq 1$, it is possible that processes are in rounds $r' > r$. $C[0]$ denotes the initial configuration. In $C[0]$, each process is in its initial state (that includes the value it proposes), all the buffers are empty and there is no message in transit.

**Structure of the proof** The structure of the proof is as in [1, 22]. The contradiction follows from the following lemmas. The first lemma shows that a configuration of $A$ after $2(t-1)$ rounds is univalent (Lemma 5). The second lemma shows that there is a configuration of $A$ that, after $2(t-1)$ rounds, is bivalent (Lemma 7). That lemma uses the fact that, assuming the existence of an algorithm $A$ that solves the binary consensus problem, there is an initial bivalent configuration (Lemma 6).

The proof does not consider all the possible runs of $A$. It relies only on the runs of $A$ in which no process crashes in odd rounds, and there is at most one process crash per even round. As the algorithm described in Figure 2 needs $2t+1$ rounds, the $2t+1$ bound proved for these runs is a tight lower bound (Theorem 4).

**Lemma 5** *Let $t < n-1$. Any configuration $C[2(t-1)]$ produced by $A$ is univalent.*

**Proof** The proof is by contradiction. Supposing that there is a configuration $C[2(t-1)]$ produced by $A$ that is bivalent, it considers two cases, namely, either there is a bivalent configuration $C[2t-1]$ that can be attained by $A$ from $C[2(t-1)]$ (case 1, Figure 4.a), or no configuration $C[2t-1]$ attained by $A$ from $C[2(t-1)]$ is bivalent (case 2, Figure 5.a).




(a) $C$ is bivalent

(b) Obtaining the contradiction

Figure 4: The configuration $C[2t-1]$ is bivalent (Lemma 5, Case 1)

Case 1. Let $C = C[2t-1]$ be a bivalent configuration attained by $A$ from $C[2(t-1)]$. As $C$ is bivalent and $A$ decides in $2t$ rounds (assumption), there are two runs $R_0$ and $R_1$ of $A$ that extend $C$ to $C_0[2t]$ and $C_1[2t]$, respectively, such that $C_0[2t]$ is 0-valent (0 is decided by the end of the round $2t$), and $C_1[2t]$ is 1-valent (1 is decided by the end of the round $2t$). (See Figure 4.) Let us notice that (1) until the configuration $C$, $R_0$ and $R_1$ are identical, and (2) during the round $2t$ of $R_0$ or $R_1$ (or both), one process crashes (otherwise, the non-faulty processes would receive the same messages during the round $2t$ and would consequently decide the same value by the end of that round in both the runs $R_0$ and $R_1$). In both the runs $R_0$ and $R_1$, the same set $P$ of processes enter the round $2t$. This follows from the fact that no process crashes during the round $2t-1$ (by assumption, no process crashes during the odd rounds). Let $k = |P|$.

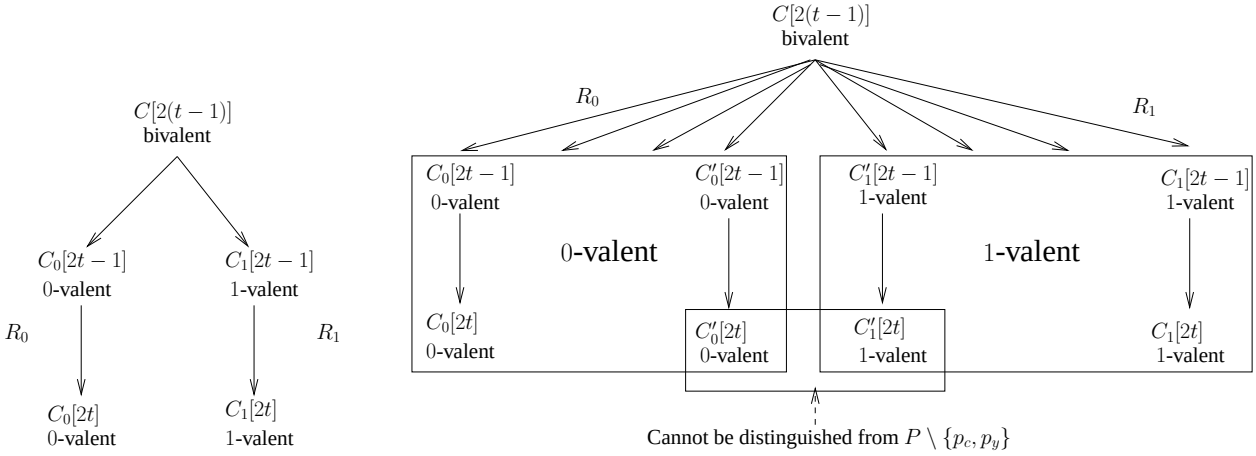
Let $P'$ be the set of processes that are correct in both the runs $R_0$ and $R_1$. We have $|P'| \geq k-2$, because there is at most one crash during the last round of $R_0$, at most one crash during the last round of $R_1$, and these crashes can be from two different processes. Moreover, as $k \geq n-(t-1)$ (from the first round until the $(2t-1)$th round, there is at most one crash per even round), and $t < n-1$ (lemma assumption), we have $k-2 \geq 1$, which means that $P'$ contains at least one process. Let $p_x$ a process of $P'$.

Let $C'_0[2t]$ (resp., $C'_1[2t]$) be a configuration similar to $C_0[2t]$ (resp., $C_1[2t]$) except for $p_x$, and $R'_0$ (resp., $R'_1$) be the corresponding run producing $C'_0[2t]$ (resp., $C'_1[2t]$) (see Figure 4.b). More precisely, during the round $2t$, all the correct processes in $C'_0[2t]$ (resp., $C'_1[2t]$) have received the same messages, but $p_x$ that has received exactly $k$ messages, one from each process that has entered the $2t$th round (as indicated before, one process has crashed during the round $2t$ in $R_0$, $R_1$ or both, but that process has sent its round $2t$ message to $p_x$ before crashing)[2]. We have the following.

- Only the state of $p_x$ can differ in $C_0[2t]$ and $C'_0[2t]$. As $t < n-1$, in addition to $p_x$, there is at least another correct process $p_y$ that decides in both the runs $R_0$ and $R'_0$ ending at $C_0[2t]$ and $C'_0[2t]$, respectively. Due to the agreement property of $A$, they decide the same value in both runs, namely 0.
- Similarly, only the state of $p_x$ can differ in $C_1[2t]$ and $C'_1[2t]$. As $t < n-1$, in addition to $p_x$, there is at least another correct process $p_z$ that decides in both the runs $R_1$ and $R'_1$ ending at $C_1[2t]$ and $C'_1[2t]$. Due to the agreement property of $A$, they decide the same value in both runs, namely 1.

[2]Let us notice that $C_0[2t]$ and $C'_0[2t]$ can be identical if $p_x$ has also received the $k$ messages in $R_0$ (and then $R_0$ and $R'_0$ also can be identical). The same can occur for $C_1[2t]$ and $C'_1[2t]$.

- Let us finally observe that $p_x$ has the same local state in $C'_0[2t]$ and $C'_1[2t]$. It follows that it decides the same value $v$ in both the runs $R'_0$ and $R'_1$. A contradiction with the two previous items, which concludes the proof of the lemma for the first case.




(a) Univalent configurations

(b) Obtaining the contradiction

Figure 5: All the configurations $C[2t-1]$ are univalent (Lemma 5, Case 2)

Case 2. No bivalent configuration $C[2t-1]$ is attained by $A$ from $C[2(t-1)]$. Starting from the bivalent configuration $C[2(t-1)]$, let us consider the run $R_0$ that extends the configuration $C[2(t-1)]$ to the configuration $C_0[2t]$ such that there is no crash between these two configurations. Let us notice that, as there is no crash, $C_0[2t]$ is unique. As $A$ decides during the $2t$th round, let us assume (without loss of generality) that $C_0[2t]$ is 0-valent. Moreover, as $C[2(t-1)]$ is bivalent, there is a run $R_1$ that extends $C[2(t-1)]$ to a 1-valent configuration $C_1[2t]$. Due to the case assumption, each of $C_0[2t]$ and $C_1[2t]$ is obtained from an intermediary univalent configuration, that we denote $C_0[2t-1]$ and $C_1[2t-1]$, respectively (see Figure 5.a).

Let us observe that, in the run $R_1$ ending in $C_1[2t]$, there is at least one crash, otherwise we would have $C_1[2t] = C_0[2t]$. Moreover, when considering additionally the pattern of failures (no crash in odd rounds, and at most one crash during each even round), we can conclude that there is exactly one crash, and the corresponding process crashes while executing the $2t$th round. Moreover, this crash occurs before the configuration $C_1[2t-1]$ (i.e., before the last alive process in round $2t-1$ enters the round $2t$), otherwise the configurations $C_0[2t-1]$ and $C_1[2t-1]$ could not be distinguished (a contradiction as one is 0-valent while the other is 1-valent).

Let $P$ be the set of processes that start the $(2t-1)$th round in $R_0$ and $R_1$ (they are the same processes as $R_0$ and $R_1$ are identical until $C[2(t-1)]$ and, by assumption, no process crashes during an odd round, here the round $(2t-1)$). Let $|P| = k$. As in the first case (same reasoning), we have $k \geq 3$. Let $p_c$ be the process that crashes while executing the last round of $R_1$.

As, after the round $2(t-1)$, the run $R_0$ has no crash, each of the $k$ processes of $P$ enters the round $2t$ after having received $k$ messages during the round $2t-1$. Differently, when the run $R_1$ attains the configuration $C_1[2t-1]$, due to the crash of $p_c$, some processes enter the round $2t$ after having received $k$ messages (let $P_k$ be this set of processes), while other processes enter the round $2t$ after having received

$k-1$ messages (let $P_{k-1}$ be this set of processes)[3]. The set $P_k \subset P$ is not empty, otherwise the same final configuration $C[2t]$ could be attained from both the configurations $C_0[2t-1]$ and $C_1[2t-1]$.

To obtain the final contradiction that will prove the lemma for the second case, let us construct "intermediate" configurations between $C_1[2t-1]$ and $C_0[2t-1]$ as follows. We iteratively move one process from $P_{k-1}$ to $P_k$ (each iteration defining a new configuration) until all the processes of $P_{k-1}$ have been moved into $P_k$. The process that is moved receives now $k$ messages, while the other processes receives $k$ or $k-1$ messages (as in the configuration from which the new configuration is obtained). In that way, we obtain a sequence of "$C[2t-1]$ configurations" that starts at the configuration $C_1[2t-1]$ and ends at the configuration $C_0[2t-1]$ (see Figure 5.b). All these configurations can be attained from $C[2(t-1)]$, and by assumption are univalent (if one of them was bivalent, we would be in Case 1). Consequently, there are two adjacent configurations $C'_0[2t-1]$ and $C'_1[2t-1]$ in this sequence such that one is 0-valent while the other is 1-valent. Moreover, these two configurations differ only in the number of message received during the $(2t-1)$th round by a process $p_y \in P \setminus \{p_c\}$. Let us recall that $P$ contains at least 3 processes, and consequently $|P \setminus \{p_y, p_c\}| \geq 1$.

Let us consider the run where, after $C'_0[2t-1]$ (resp., $C'_1[2t-1]$), all the processes in $P \setminus \{p_c, p_y\}$ (there is at least one such process) receives the $k-1$ messages from the processes in $P \setminus \{p_y\}$. This run ends in the final configuration $C'_0[2t]$ (resp., $C'_1[2t]$). As $C'_0[2t-1]$ is 0-valent, $C'_0[2t]$ is 0-valent. Similarly, as $C'_1[2t-1]$ is 1-valent, $C'_1[2t]$ is 1-valent. But the processes in the set $P \setminus \{p_y, p_c\}$ cannot distinguish $C'_0[2t]$ from $C'_1[2t]$. Consequently, these two configurations have the same valence, a contradiction which completes the proof of the lemma for the second case. $\square_{Lemma\ 5}$

**Lemma 6** *There is a bivalent initial configuration.*

**Proof** (The proof is close to the corresponding proof given in [22]). Due to the very existence of $A$ and the fact that it satisfies the consensus validity property, the configuration with only zeroes (ones) is trivially 0-valent (1-valent). Let us suppose that all the initial configurations are univalent. Clearly, there are two initial configurations $C_0$ and $C_1$ that differ in the value proposed by only one process (say $p_x$) such that $C_0$ is 0-valent, while $C_1$ is 1-valent.

As indicated before, this lemma is used in the proof of Lemma 7. As that lemma considers only the runs in which no process crashes in the odd rounds and at most one process crashes at each even round, this pattern of failures is also considered in the current proof. Let us consider the configuration $C[1]$ attained by the algorithm $A$ in the following run that satisfies the previous crash pattern.

When it executes the first round, the process $p_x$ receives a message from each process, proceeds to the second round, and crashes before sending any message during the second round. On the other side, each other process $p_i$ is informed of the crash of $p_x$ while it is waiting for messages sent during the first round and terminates this round without receiving the message from $p_x$. As, no process $p_i \neq p_x$ ever receives a message from $p_x$, the configuration $C[1]$ can be attained from any of the initial configuration $C_0$ or $C_1$. Hence, $C[1]$ is bivalent, and consequently both $C_0$ and $C_1$ have to be bivalent, which contradicts the initial assumption. $\square_{Lemma\ 6}$

**Lemma 7** *Let $t < n-1$. There is a configuration $C[2(t-1)]$ produced by $A$ that is bivalent.*

**Proof** The proof is by induction. Lemma 6 has proved the base case, namely, given a binary consensus algorithm $A$, there is an initial bivalent configuration. Let us assume that there is a bivalent configuration

[3]This is due to a misleading notification of the crash of $p_c$. Moreover, let us notice that, due to asynchrony, the missing message is not necessarily the message from $p_c$.

$C[2(\ell-1)]$ for some $\ell$ such that $1 \leq \ell \leq t-1$. We show that there is a configuration $C[2\ell]$ produced by $A$ that is bivalent.

As for the previous lemmas, the proof that there is a bivalent configuration $C[2\ell]$ is by contradiction, so we assume that all the configurations $C[2\ell]$ attained by $A$ are univalent. The proof considers two cases: there is a configuration $C[2\ell-1]$ attained from $C[2(\ell-1)]$ that is bivalent (case 1), or no configuration $C[2\ell-1]$ attained from $C[2(\ell-1)]$ is bivalent (case 2). The proof of each case resembles the proof of a case in Lemma 5. The main difference lies in the fact that the reasoning can no longer use the fact that $C[2\ell]$ is a final configuration. As in Lemma 6 (that considered the case of $C[0]$), configurations that follow $C[2\ell]$ have to be considered.

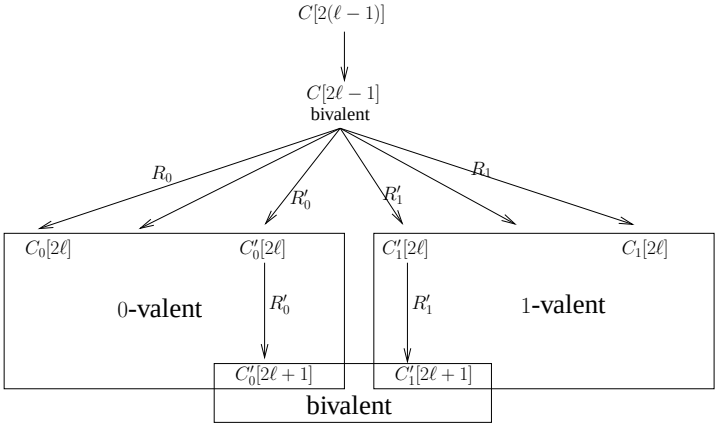



Figure 6: The configuration $C[2\ell-1]$ is bivalent (Lemma 7, Case 1)

Case 1. There is a configuration $C[2\ell-1]$ attained from $C[2(\ell-1)]$ that is bivalent (see Figure 6). Considering that configuration, let us consider two runs $R_0$ and $R_1$, identical until $C[2\ell-1]$, and such that $R_0$ attains the configuration $C_0[2\ell]$ that is 0-valent, while $R_1$ attains the configuration $C_1[2\ell]$ that is 1-valent ($C_0[2\ell]$ and $C_1[2\ell]$ are univalent by assumption). Moreover, without loss of generality, let us assume that no process crashes in $R_0$ between $C[2\ell-1]$ and $C[2\ell]$, while a process $p_c$ crashes in $R_1$ while it executes the round $2\ell$. Let us notice that, due to asynchrony, the crash of $p_c$ can appear in the configuration $C[2\ell-1]$, and be notified to some processes before this configuration is attained, and after it for other processes.

The reasoning is now similar as the one used in case 2 of the proof of Lemma 5[4]. Let $P$ be the set of processes that started the round $2\ell$. They are the same processes that start this round in $R_0$ and $R_1$. Let $k = |P|$. Due to $\ell \leq t-1$, $t < n-1$, and the assumption that no process crashes while executing an odd round and at most one process crashes per even round, we have $k \geq 4$ (as at most $\ell-1$ processes crash during the first $2\ell-1$ rounds, $n-(\ell-1) \geq n-t+2 \geq 4$ processes start the round $2\ell$). In $R_0$, each of $k$ processes $p_i$ that enters the round $2\ell$ receives a set of $k$ or $k-1$ messages (this difference is due the misleading notification of the crash of $p_c$). Let $P_i^0$ be this set of messages. Similarly, let $P_j^1$ be the set of $k$ or $k-1$ messages received by each process $p_j$ (but $p_c$) that execute the round $2\ell$ in $R_1$.

Similarly to the proof of Lemma 5, it is possible to construct a sequence of univalent configurations starting from $C_1[2\ell]$ and ending at $C_0[2\ell]$ including two adjacent configurations $C'_1[2\ell]$ and $C'_0[2\ell]$ such that (1) $C'_1[2\ell]$ is 1-valent, $C'_0[2\ell]$ is 0-valent, (2) they differ only in the set of messages received by some process $p_y \in P \setminus \{p_c\}$, and (3) they cannot be distinguished by the processes in $P \setminus \{p_c, p_y\}$ (see Figure 6).

Now, because the configurations $C'_1[2\ell]$ and $C'_0[2\ell]$ are not final configurations, the reasoning is similar to the one used in Lemma 6. Let $R'_0$ (resp., $R'_1$) be a run such that from $C'_0[2\ell]$ (resp., $C'_1[2\ell]$):

[4]While the reasoning in Lemma 5 is based on the *number* $k$ or $k-1$ of messages received by a process during the round $2t$, here the reasoning is based on the *set* of the messages received by a process during the round $2\ell$.

1. $p_y$ receives a message from each process that started the round $2\ell + 1$ (those are the processes in $P \setminus \{p_c\}$), and then crashes just after entering the round $2\ell + 2$ and before sending any message (let us notice that $p_y$ can be crashed only in an even round),
2. Each other process is informed of the crash of $p_y$ while it is waiting for messages sent during the round $2\ell + 1$, and terminates its current round $2\ell + 1$ without receiving the message from $p_y$.

As $p_y$ has not sent a round $2\ell+2$ message, it follows that no alive process can distinguish $C'_0[2\ell+1]$ and $C'_1[2\ell+1]$. It follows that both 0 and 1 can be decided from $C'_0[2\ell+1]$ and $C'_1[2\ell+1]$, i.e., they are bivalent, which contradicts the assumption they are univalent. This completes the proof of the lemma for the first case.

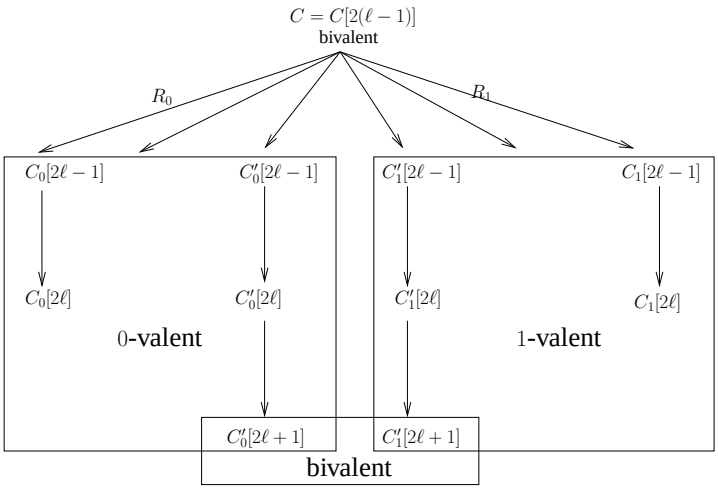



Figure 7: No configuration $C[2\ell - 1]$ is bivalent (Lemma 7, Case 2)

Case 2. No configuration $C[2\ell - 1]$ attained from $C[2(\ell - 1)]$ is bivalent. The structure of the proof and its underlying design are depicted implicitly in Figure 7. The associated reasoning, is nearly the same as in the previous cases, and is left to the reader. $\square_{Lemma\ 7}$

**Theorem 3** *Let $t < n - 1$. There is no consensus algorithm that always terminates in at most $2t$ rounds in the $\mathcal{AARS}_{n,t}[\psi]$ model.*

**Proof** The proof is an immediate consequence of the Lemmas 5 and 7. $\square_{Theorem\ 3}$

The following theorem is an immediate consequence of the previous theorem and Theorem 2.

**Theorem 4** *Let $t < n - 1$. The algorithm described in Figure 2 is optimal (for the number of rounds) in the $\mathcal{AARS}_{n,t}[\psi]$ model.*

# 5 Early decision and halting

## 5.1 Early decision

The aim is here to allow the processes to decide before the round $2t + 1$ when there are few failures. Let $f$ ($0 \leq f \leq t$) be the actual number of faulty processes. The corresponding consensus lower bound is $\min(t+1, f+2)$ rounds in synchronous systems [8, 31, 34, 41]. What is the lower bound in $\mathcal{AARS}^{cl}_{n,t}[\psi]$?

Compared to synchronous systems, the new difficulty we have to cope with in $\mathcal{AARS}^{cl}_{n,t}[\psi]$ lies in the fact that, due misleading messages, during a round a process can miss messages from processes that have not crashed. Providing early decision in such a context is a real challenge. Our intuition is that early decision in $\mathcal{AARS}^{cl}_{n,t}[\psi]$ requires the processes to decide simultaneously during the very same round. The simultaneous agreement problem, introduced in [17, 19], has been shown to be strongly related to the "common knowledge" theory [26], and has received some attention in the literature (e.g., [35, 36]). So, we conjecture that early decision and halting in $\mathcal{AARS}^{cl}_{n,t}[\psi]$ requires simultaneous agreement and should be attained in $2t+1-D$ rounds where $D$ ($0 \leq D \leq t$) is parameter defined from the actual failure pattern [19]. A first step in that direction is done in Appendix C where it is shown that, in $\mathcal{AARS}^{cl}_{n,t}[\psi]$, simultaneous decision by round $t+1$ is impossible when $f = 0$.

### 5.2 The system model $\mathcal{AARS}^{op}_{n,t}[\psi]$

This paper addresses early decision in a model, denoted $\mathcal{AARS}^{op}_{n,t}[\psi]$(where $op$ stands for $op$en), derived from, and less constraining than, $\mathcal{AARS}^{cl}_{n,t}[\psi]$. This model is round-based but not round communication-closed. During any round $r$, in addition to the messages tagged $r$, a process can send or receive and process a round-free message, i.e., a message that is not tagged by a round number. This model allows the behavior of a process to be defined by two tasks: a round-based task $T1$, and a task $T2$ that processes the round free messages. The model $\mathcal{AARS}^{op}_{n,t}[\psi]$ assumes also that each process knows initially $n$ and $t$.

It is interesting to recall that, differently from what can be done in the round-based synchronous model, a lot of "round-based" asynchronous algorithms do actually assume a model similar to $\mathcal{AARS}^{op}_{n,t}[\psi]$. This is the case, for example, of the round-based consensus algorithms that assume an underlying failure detector such as the eventual leader $\Omega$. Before deciding, a process broadcasts a DECIDE() that allows its receiver to stop executing its round-based task, and decide immediately (e.g., [11, 12, 39]).

### 5.3 An early deciding algorithm for $\mathcal{AARS}^{op}_{n,t}[\psi]$

An algorithm that solves the consensus problem in $\min(2t+1, 2f+2)$ rounds in the $\mathcal{AARS}^{op}_{n,t}[\psi]$ model is described in Figure 8. As announced, it is made up of two tasks. The task $T2$ is to prevent deadlock: when a process early decides (line N5), it broadcasts a round-free DECIDE() message and, if a process $p_i$ has not yet decided when it receives such a message, it forwards it and returns the decided value (and stops accordingly).

The main task $T1$ is a round-based task partly similar to the the behavior described in Figure 2. The lines common to both algorithms have the same number. $M$ is appended to the number of a line that is modified, while the new lines are numbered N1, N2, etc.

Each process $p_i$ manages the following additional local variables: $early_i$ initialized to $false$ (its meaning will be explained later), $rec_i$ that counts the number of messages received during the current round (line N1), and a variable $k$ whose current value is such that $r_i = 2k+1$ in odd rounds and $r_i = 2k+2$ in even rounds (line N2). Moreover, a round message now carries the additional boolean value $early_i$ (line 04M).

The core of the early decision is at lines N3-N6, namely a process $p_i$ early decides during the round $r$ if the following round-dependent predicate is satisfied: the round is even, exactly $n-k = n - \lfloor \frac{r-1}{2} \rfloor$ messages EST$(r,-,-)$ have been received and each carries the value $true$ (lines N3-N4). As we will see in the proof, when satisfied, this locally evaluable predicate says that $p_i$ knows (1) the minimal value ($v$) of the $est_j$ variables of the set of the processes $p_j$ that started the round $2k+1$, and (2) that all the processes $p_j$ that started the round $2k+2$, know that value $v$. It follows that the $est_j$ values of all the processes that started the round $2k+2$ are equal to $v$, and consequently no other value can be decided. The boolean $early_j$ is used by a process $p_j$ to indicate (line 04) if during an odd round $r = 2k+1$, it has received $n-k = n - \lfloor \frac{r-1}{2} \rfloor$ round $r$ messages (line N6).

```
operation propose(v_i):
  task T1:
(01M) est_i ← v_i; r_i ← 1; early_i ← false;
(02)  while (r_i ≤ 2t + 1) do
(03)     begin asynchronous round
(04M)    brodcast EST(r_i, est_i, early_i);
(05)     wait until ( aal_i messages EST(r_i, −, −) have been received );
(06)     est_i ← min(est values received at the previous line);
(N1)     let rec_i[r_i] = number of messages received at the previous line;
(N2)     let k = ⌊(r_i − 1)/2⌋;
(N3)     if   (r_i is even) ∧ (rec_i[r_i] = n − k)
(N4)        ∧(each EST(r_i, −, early) received is such that early = true)
(N5)              then brodcast DECIDE(est_i); return(est_i) end if;
(N6)     if (r_i + 1 is even) then early_i ← (rec_i[r_i] = n − k) end if;
(07)     r_i ← r_i + 1
(08)     end asynchronous round
(09)  end while;
(10)  return(est_i).
=============================================================
  task T2: when DECIDE(est) is received do brodcast DECIDE(est); return(est) end do.
```

Figure 8: Early deciding anonymous consensus in $\mathcal{AARS}^{op}_{n,t}[\psi]$ ($n$ and $t$ are known)

## 5.4 Proof of the early deciding algorithm

**Definition** Let $EST[r]$ be the set of the values in the variables $est_i$ of the processes that enter the round $r$. Let us observe that $r1 > r2 \Rightarrow EST[r1] \subseteq EST[r2]$. Moreover, we say "$p_i$ knows $v$" at the end of a round, if $est_i = v$ at the end of that round.

**Lemma 8** *A decided value is a proposed value.*

**Proof** The proof is the same as the one of Lemma 1. $\square_{Lemma\ 8}$

**Lemma 9** *Let $0 \leq k \leq t-1$. This lemma consists of two propositions depending on the parity of the round.*

- $(i)$ *If a process $p$ receives $n-k$ messages $\text{EST}(2k+1,-,-)$ during a round $2k+1$, it knows $\min(EST[2k+1])$ at the end of the round $2k+1$.*
- $(ii)$ *If a process $p$ receives $n-k$ messages $\text{EST}(2k+2,-,true)$ during a round $2k+2$, it knows that all processes that have started the round $2k+1$ knew $\min(EST[2k+1])$ at the end of the round $2k+1$.*

**Proof** The proof is by induction on $k$.

**Base step ($k = 0$).** For the rounds $r = 1$ and $r = 2$ the result is trivial. If a process receives $n$ messages during the first round, it knows that it has received a message from each process and consequently it knows all the values in $EST[1]$. If a process receives $n$ messages $\text{EST}(2,-,true)$ during the second round, it knows that all processes know all the values in $EST[1]$ (due to the boolean $early$ which is true for all processes).

**Induction step.** Let $k > 0$. Assuming that the lemma is true for any $k' < k$ we have to show that it is also true for $k$, i.e. for the next pair of rounds $(2k+1, 2k+2)$.

- Proof of $(i)$. Let $p$ be a process that receives $n-k$ messages $\text{EST}(2k+1,-,-)$ during the round $2k+1$. Let $S$ be the number of processes that have started the round $2k+1$. Let us notice that, since $p$ receives $n-k$ messages during the round $2k+1$, we have $S \geq n-k$. The proof considers two cases according to the value of $S$.
  - $S = n-k$. In that case, $p$ has received a message from each process that have started the round $2k+1$, i.e., $p$ knows $EST[2k+1]$. Hence, it knows $\min(EST[2k+1])$.
  - $S > n-k$. We claim that there is a $k' < k$ such that:
    1. There are $n-k'$ processes that started the round $2k'+1$,
    2. There are $n-k'$ processes that started the round $2k'+2$,
    3. There are $n-k'$ processes that started the round $2k'+3$.

    This claim implies that the $n-k'$ processes that have started the round $2k'+1$ have received $n-k'$ messages $\text{EST}(,2k'+1,-,-)$ (since there is neither crashes nor decision in the rounds $2k'+1$ and $2k'+2$)[5]. Due to the item $(i)$ of the induction assumption, it follows that $p$ knows $\min(EST[2k'+1])$ at the end of round $2k'+1$. Since (1) $p$ keeps this minimum in $est_i$, and (2) $2k+1 > 2k'+1 \Rightarrow EST[2k+1] \subseteq EST[2k'+1]$, it follows that $p$ cannot receive in the future a value smaller than $\min(EST[2k'+1])$. Hence, $\min(EST[2k'+1]) = \min(EST[2k+1])$, which proves item $(i)$ of the lemma for that case.

    The proof of the claim is by contradiction. Suppose that there is no such $k' < k$. This implies that at most $n-1$ processes have started the round 3 (otherwise $k'=0$ would be such a "good" $k'$). It implies also that at most $n-2$ processes have started the round 5 (otherwise $k'=1$ would be such a "good" $k'$). More generally it implies that at most $n-x$ processes have started the round $2x+1$ (otherwise $k' = x-1$ would be such a "good" $k'$). Taking $x=k$ implies that at most $n-k$ processes have started the round $2k+1$, which contradicts the case assumption $S > n-k$, and concludes the proof of the the claim.

- Proof of $(ii)$. Let $p$ be a process that receives $n-k$ messages $\text{EST}(2k+2,-,true)$ during the round $2k+2$. Let $S$ be the number of processes that start the round $2k+1$. Let us notice that, since $p$ receives $n-k$ messages during the round $2k+2$, we have $S \geq n-k$. As previously, the proof considers two cases according to the values of $S$.
  - $S = n-k$. In that case, $p$ has received in the round $2k+2$ a message from each process that started the round $2k+1$. If all these messages contains the boolean $early$ equals to $true$, it means that each of these $n-k$ processes has received $n-k$ messages in round $2k+1$, which (due to $(i)$) implies that all of them know $\min(EST[2k+1])$.
  - $S > n-k$. The claim stated in the proof of item $(i)$ is still correct. That claim implies that the $n-k'$ processes that have started the round $2k'+1$ have received $n-k$ messages (since there is neither a crash nor decision in the rounds $2k'+1$ and $2k'+2$, see footnote 5). It follows by induction that all these processes know $\min(EST[2k'+1])$ at the end of round $2k'+1$, and as $EST[2k'+1] \subseteq EST[2k+2]$, this remains true at the end of the round $2k+2$.

$\square_{Lemma\ 9}$

**Lemma 10** *No two processes decide different values.*

[5] This follows from the following observation. If $X$ processes start the round $r$ and $X$ processes start the round $r+1$, then no process has crashed or decided during the round $r$. Taking $X = n-k'$ and the round $r = 2k'+1$, and $X = n-k'$ and the round $r = 2k'+2$, shows that there are neither crashes no decision during the rounds $2k'+1$ and $2k'+2$.

**Proof** If no process decides at line N5, the algorithm behaves as the non-early deciding algorithm described in Figure 2, and then the agreement property follows from Lemma 4. So, the rest of the proof considers only the case where a process decides at line N5. Let $r$ be the first round during which a process (say $p_i$) decides at line N5. We show that the estimate value $est_j$ of all the processes that terminate the round $r$ are equal. Hence, the processes that at line N5 of $r$ decides the same value $v$, and the processes that progress to $r+1$ have $v$ as estimate value, which proves the agreement property.

As $p_i$ decides during $r$ at line N5, the early termination predicate of lines N3-N4 is satisfied, i.e., $r$ is even (say $r = 2k+2$), and $p_i$ has received exactly $n-k$ EST$(2k+2, -, early)$ messages, and each of these messages is such that $early = true$. It follows from the item $(ii)$ of Lemma 9 that all the processes $p_j$ that terminated the round $2k+1$, have $est_j = \min(EST[2k+1])$ by the end of the round $2k+1$, which completes the proof of the lemma. $\square_{Lemma\ 10}$

**Lemma 11** *Let $f$ denote the actual number of process crashes. We have $(i)$ each correct process decides, and $(ii)$ no process decides in more than $\min(2f+2, 2t+1)$ rounds.*

**Proof** The proof considers each item separately.

- Proof of $(i)$. If no process executes line N5, the proof of Lemma 2 applies, and all the correct processes decide in $2t+1$ rounds. If a process executes line N5, the item $(i)$ follows due to the DECIDE() messages send by that process.
- Proof of $(ii)$. If $f = t$, due to item $(i)$ all processes decide and, as there are at most $2t+1$ rounds, the item $(ii)$ follows. So, let us assume $f < t$ and a process (say $p_i$) starts the round $2f+3$. We show a contradiction. As it has not decided by round $2f$, it follows from the early decision predicate evaluated at the lines N3-N4 that:
  - $(rec_i[2] \neq n) \vee (\exists\, j : rec_j[1] \neq n)$ (as $p_i$ does not decide during the round 2),
  - $(rec_i[4] \neq n-1) \vee (\exists\, j : rec_j[3] \neq n-1)$ (as $p_i$ does not decide during the round 4), etc.,
  - $(rec_i[2f] \neq n-f) \vee (\exists\, j : rec_j[2f-1] \neq n-f)$ (as $p_i$ does not decide during the round $2f$).

  It follows from these items (and the safety of the failure detector $\psi$) that $n-f$ processes have crashed by the end of the round $2f$, from which we conclude that no more crash occurs during the round $2f+1$ and $2f+2$. During these two rounds, each of the $n-f$ correct processes receives $n-f$ EST$(2f+1, -, -)$ messages and $n-f$ EST$(2f+2, -, true)$ messages. The predicate of lines N3-N4 is then satisfied for each correct process, which proves the lemma.

  $\square_{Lemma\ 11}$

**Theorem 5** *The algorithm described in Figure 8 solves the consensus problem in $\min(2f+2, 2t+1)$ rounds in the $\mathcal{AARS}^{op}_{n,t}[\psi]$ model (where $f$ denotes the actual number of process crashes).*

**Proof** The proof follows from the Lemmas 8, 10 and 11. $\square_{Theorem\ 5}$

## 6 From consensus to $k$-set agreement

This section considers the $k$-set agreement problem in anonymous asynchronous crash-prone message passing systems.

**The $k$-set agreement problem** The $k$-set agreement problem has been introduced in [13] to study how the number of choices ($k$) allowed to the processes is related to the maximum number of faulty processes ($t$). It is defined by the same validity and termination properties as the consensus problem, and the following agreement property: at most $k$ different values can be decided (so, consensus is 1-set agreement). The $k$-set agreement problem cannot be solved in non-anonymous asynchronous crash-prone systems as soon as $k \leq t$ [9, 30, 42]. Differently, it can always be solved in round-based synchronous systems where $\lfloor \frac{t}{k} \rfloor + 1$ is a lower bound on the number of rounds [14].

### 6.1 Solving $k$-set agreement in $\mathcal{AARS}^{cl}_{n,t}[\psi]$ with $t \leq n - k$

The algorithm described in Figure 2, where $2t + 1$ is replaced by $2 \lfloor \frac{t}{k} \rfloor + 1$ solves the $k$-set agreement in $\mathcal{AARS}^{cl}_{n,t}[\psi]$. The proof of the validity and termination properties are the same as their consensus counterparts. So, we consider here only the proof of the agreement property. The assumption $t \leq n - k$ generalizes the assumption $t \leq n - 1$ associated with the consensus problem.

**Lemma 12** *Let $t \leq n - k$. If at most $k - 1$ processes crash during two consecutive rounds $r$ and $r + 1$, then the set of the estimates of the processes that terminate the round $r + 1$ contains at most $k$ different values at the end of $r + 1$.*

**Proof** The proof is the proof of Lemma 14 where $\ell$ is replaced by 1. $\square_{Lemma\ 12}$

**Lemma 13** *Let $t \leq n - k$. At most $k$ different values are decided (agreement).*

**Proof** As previously, the proof is the proof of Lemma 15 where $\ell$ is replaced by 1. $\square_{Lemma\ 13}$

**Theorem 6** *The algorithm described in Figure 2 (where $(2t + 1)$ is replaced by $2 \lfloor \frac{t}{k} \rfloor + 1$) solves the $k$-set agreement problem in $2 \lfloor \frac{t}{k} \rfloor + 1$ rounds in the $\mathcal{AARS}^{cl}_{n,t}[\psi]$ model where $t \leq n - k$.*

**Proof** The proof follows from the lemmas 1, 2, and 13. $\square_{Theorem\ 6}$

### 6.2 Solving the $k$-set agreement with weaker failure detectors

**The failure detector class $\psi_\ell$** As, when $k > 1$, the $k$-set agreement problem is weaker than consensus, it should be possible to use a failure detector weaker than $\psi$ in order to solve it. So, let us consider the class of failure detectors, denoted $\psi_\ell$, $1 \leq \ell \leq n$, that is a simple generalization of $\psi$. It is defined as follows (the notation is the same as in Section 2.2):

- Safety: $\forall \tau :\ aa\ell_i^\tau \geq n - f^\tau - (\ell - 1)$.
- Liveness: $\exists \tau :\ \forall \tau' \geq \tau : n - f - (\ell - 1) \leq aa\ell_i^{\tau'} \leq n - f$.

From this definition, we obtain a family of failure detector classes $\{\psi_\ell\}_{1 \leq \ell \leq n}$. It is easy to see that $\psi_1$ is $\psi$ and $\psi_\ell$ is weaker than $\psi_{\ell-1}$.

**A $k$-set algorithm for $\mathcal{AARS}^{cl}_{n,t}[\psi_\ell]$** Interestingly, when the number of rounds $2t+1$ is replaced by $2\lfloor\frac{t}{k-\ell+1}\rfloor+1$, the algorithm described in Figure 2 solves the $k$-set agreement problem in $\mathcal{AARS}^{cl}_{n,t}[\psi_\ell]$ (assuming $t<n-k+\ell$ and $\ell\leq k$). As we can see, the $\psi$-based consensus algorithm described in Figure 2 and its $\psi$-based $k$-set agreement variant (described in the previous section), are two particular instances of the general $\psi_\ell$-based algorithm. These instances consider $\ell=1$, i.e., the strongest class in the failure detector family $\{\psi_\ell\}_{1\leq\ell\leq n}$.

As previously, the proof of the validity and termination properties are the same as their consensus counterparts. So, we consider here only the proof of the agreement property (that has the same structure as in the consensus case). Let us recall (Definition 1) that a processes *terminates* a round $r$ if it proceeds to $r+1$ or decides if $r$ is the last round it executes.

**Lemma 14** *Let $t\leq n-k$ and $1\leq\ell\leq k$. If at most $k-\ell$ processes crash during two consecutive rounds $r$ and $r+1$, then the set of the estimates of the processes that terminate the round $r+1$ contains at most $k$ different values at the end of $r+1$.*

**Proof** Let $r$ and $r+1$ be two consecutive rounds with at most $k-\ell$ crashes (as $\ell\leq k$, we have $k-\ell\geq 0$), and $P_r$ the set of processes that enter the round $r$ (whatever the time they enter it). Let $p$ be the first process that terminates the round $r+1$ and $\tau$ the corresponding time. Finally, let $Q_{r+1}$ be the subset of $P_r$ that contains the processes that have entered the round $r+1$ at time $\tau$, and considering a process $p_i$ that terminates the round $r+1$, let $aa\ell(i,r+1)$ be the number of round $r+1$ messages that allow $p_i$ to terminate that round.

To terminate the round $r+1$, the process $p$ has received (at the latest at time $\tau$) $aa\ell(i,r+1)$ round $r+1$ messages, from which we conclude that at least $aa\ell(i,r+1)$ processes has entered the round $r+1$ at time $\tau$. Moreover, $aa\ell(i,r+1)\geq|P_r|-(k-\ell)-(\ell-1)=|P_r|-(k-1)$,[6] from which follows that at most $k-1$ processes have not yet entered the round $r+1$ at time $\tau$ (those are the processes in $P_r\setminus Q_{r+1}$). The reasoning is now made up of two steps.

1. During the round $r$, each process in $Q_{r+1}$ has received between $|P_r|-(k-1)$ and $|P_r|$ round $r$ messages in order to to enter the round $r+1$. Consequently, these processes have at most $k$ different estimate values which are amongst the $k$ smallest estimates of the processes in $P_r$.
2. The processes in $P_r\setminus Q_{r+1}$ eventually enter the round $r+1$ (or crash before entering it). When they enter the round $r+1$, there may be additional crashes, and consequently these processes may enter the round $r+1$ with an estimate that does not belong to the $k$ smallest estimates of the processes in $P_r$. However, as $t\leq n-k$, we have $t+k\leq n$ which means that there are at least $k$ correct processes.

   It follows that, in order to terminate the round $r+1$, each process $q$ in $P_r\setminus Q_{r+1}$ (that does not crash when it is in $r$ or $r+1$) receives at least $k$ round $r+1$ messages. As $|P_r\setminus Q_{r+1}|\leq k-1$, it follows that $q$ receives at least one round $r+1$ message from a process in the set $Q_{r+1}$. Consequently it updates its estimate to one of the $k$ smallest estimates of the processes in $P_r$. Hence, any process that terminates the round $r+1$ is such that its current estimate is one of the $k$ smallest estimates of the processes in $P_r$.

$\square_{Lemma\ 14}$

**Lemma 15** *Let $t\leq n-k$ and $1\leq\ell\leq k$. At most different values are decided (agreement).*

**Proof** The proof is nearly the same as its consensus counterpart. There are two cases.

[6] In the lower bound of $aa\ell(i,r+1)$, the quantity $(\ell-1)$ corresponds to the maximal mistake that the output $aa\ell_i$ of $p_i$'s failure detector can make (as $\ell\geq 1$, this quantity cannot be negative), while the quantity $(k-\ell)$ corresponds to the maximal number of crashes during the rounds $r$ and $r+1$ (by assumption $k-\ell\geq 0$).

- Case 1. In the sequence of $2\lfloor t/(k-\ell+1)\rfloor+1$ rounds, there are two consecutive rounds with at most $k-\ell$ crashes. Let $r$ and $r+1$ be these two rounds, with $r \leq 2\lfloor t/(k-\ell+1)\rfloor$. It follows from Lemma 12 that all the processes that terminate the round $r+1$ have at most $k$ different estimates. Hence, at most $k$ different values can be decided at the end of the round $2\lfloor t/(k-\ell+1)\rfloor+1$.
- Case 2. In the sequence of $2\lfloor t/(k-\ell+1)\rfloor+1$ rounds, there are no two consecutive rounds with less than $k-\ell+1$ crashes. Let us observe that, in that case, the $2\lfloor t/(k-\ell+1)\rfloor$. first rounds are such that they suffer at least $(k-\ell+1)\lfloor t/(k-\ell+1)\rfloor$ crashes. Hence, the last round can contain at most $t-(k-\ell+1)\lfloor t/(k-\ell+1)\rfloor$ crashes. As $t-(k-\ell+1)\lfloor t/(k-\ell+1)\rfloor < (k-\ell+1)$, it follows that at most $k$ different estimates can be computed during the last round (since each process that enter in the last round misses at most $(k-\ell)$ messages due to crashes and at most $(\ell-1)$ messages due to behavior of the failure detector), which completes the proof of the lemma.

$\square_{Lemma\ 15}$

**Theorem 7** *Let $k \leq t \leq n-k$ and $1 \leq \ell \leq k$. The algorithm described in Figure 2,where $(2t+1)$ is replaced by $2\left\lfloor \frac{t}{k-\ell+1}\right\rfloor+1$, solves the $k$-set agreement problem in $2\left\lfloor \frac{t}{k-\ell+1}\right\rfloor+1$ rounds in the $\mathcal{AARS}^{cl}_{n,t}[\psi_\ell]$ model.*

**Proof** The proof follows from the lemmas 1, 2, and 15. $\square_{Theorem\ 7}$

**Discussion** Let us consider the instance of the general $\psi_\ell$-based algorithm where the number of rounds is fixed to a predetermined value $R$ (instead of $2\lfloor (t/k-\ell+1)\rfloor+1$.

- Then, that algorithm instance solves the $k$-set agreement problem where $k$ is the smallest value such that $R \geq 2\lfloor t/(k-\ell+1)\rfloor+1$.
- From a different point of view, the weakest failure detector class $\psi_\ell$ for which that instance can solve the $k$-set agreement problem in $R$ rounds is defined by the greatest value of $\ell$ such that $R \geq 2\lfloor t/(k-\ell+1)\rfloor+1$ (if such a value does exist[7]).

This clearly shows how the algorithm captures and links its cost (measured by its time complexity $R$), the power of the failure detector the system is equipped with (this power is defined by $\ell$, the greater $\ell$, the weaker the power of the underlying failure detector), and the difficulty of the considered set agreement problem (measured by the coordination degree $k$: $k'$-set agreement is more difficult than $k$-set agreement if $k'<k$). Solving a more difficult problem requires either more rounds, or a more powerful failure detector class than solving an easier problem. In the $\mathcal{AARS}_{n,t}[\psi_\ell]$ model, the three critical parameters $R$, $k$ and $\ell$ are related by the simple formula $R=2\lfloor\frac{t}{k-\ell+1}\rfloor+1$.

# 7 Conclusion

This paper has investigated the consensus problem in asynchronous systems where the processes are (1) prone to crash and (2) anonymous. Due to the impossibility of solving consensus in presence of crashes and asynchrony only, the anonymous system has to be enriched with a failure detector strong enough to face these three adversaries (asynchrony, failures and anonymity). The proposed class of failure detectors (denoted $\psi$) provides each process with an upper bound on the number of alive processes. (Such a failure

[7] Let us notice that there are cases where such an integer $\ell$ does not exist. As an example, let us take $t=3$, $k=1$ and $R=2$. It is easy to see that there is no positive value for $\ell$ such that $R=2\geq 2\lfloor\frac{3}{1-\ell+1}\rfloor+1$. This means that to solve consensus ($k=1$) in $\mathcal{AARS}^{cl}_{n,t}[\psi_\ell]$, we need $\ell=1$ and this entails $R=2\lfloor\frac{3}{1}\rfloor+1=7$ rounds.

detector is the counterpart of a perfect failure detector in an asynchronous non-anonymous system). After having shown that it is possible to solve consensus in such a system in $2t+1$ rounds, the paper has presented one of its main results, namely the proof that no $\psi$-based algorithm can solve consensus in less than $2t+1$ rounds. An early-deciding consensus algorithm has then been presented in which the processes decide in at most $\min(2f+2, 2t+1)$ rounds. This makes us inclined to think that, when comparing to asynchronous non-anonymous system enriched with a perfect failure detector (in which case consensus can be solved in $\min(t+1, f+2)$ rounds), anonymity doubles the price. It is conjectured that $\min(2f+2, 2t+1)$ rounds is the early deciding lower bound.

Then the paper has generalized the previous results to the $k$-set agreement problem. To that end, it has considered a weakened family of failure detector classes, denoted $\{\psi_\ell\}_{1\leq\ell\leq k}$, and has shown that $k$-set agreement can be solved in $R_{t,\ell} = 2\lfloor \frac{t}{k-\ell+1} \rfloor + 1$ asynchronous rounds despite anonymity. As $\psi_1$ is $\psi$, this means that $2\lfloor \frac{t}{k} \rfloor + 1$ rounds are sufficient in systems equipped with $\psi$.

This work leaves open problems for future research. Among them there are the following ones.

- Design a simultaneous consensus algorithm in the $\mathcal{AARS}^{cl}_{n,t}[\psi]$ model.
- Prove (or disprove) that $\min(2f+2, 2t+1)$ rounds is the lower bound for early decision in the $\mathcal{AARS}^{op}_{n,t}[\psi]$ model.
- Investigate the question of the weakest failure detector class for solving consensus despite asynchrony, anonymity and failures. (An introductory view of this problem appears in Appendix B.
- Assuming $k < t \leq n-k$, show (or disprove) that there is a $\psi_\ell$-based $k$-set agreement algorithm in the $\mathcal{AARS}^{cl}_{n,t}[\psi]$ model if and only if $1 \leq \ell \leq k$.
- Design an early deciding $k$-set agreement algorithm for the $\mathcal{AARS}^{op}_{n,t}[\psi]$ model.

# References


[1] Aguilera M.K. and Toueg S., A Simple Bivalency Proof that $t$-Resilient Consensus Requires $t+1$ Rounds. *Information Processing Letters*, 71:155-178, 1999.

[2] Angluin D., Local and Global Properties in Networks of Processes. *Proc. 12th Symposium on Theory of Computing (STOC'80)*, ACM Press, pp. 82-93, 1980.

[3] Angluin D., Aspnes J., Diamadi Z., Fischer M.J. and Peralta R., Computation in Networks of Passively Mobile Finite-state Sensors. *Distributed Computing*, 18(4):235-253, 2006.

[4] Aspnes J., Fich Ellen F. and Ruppert E., Relationship between Broadcast and Shared Memory in Reliable Anonymous Distributed Systems. *Distributed Computing*, 18(3):209-219, 2006.

[5] Aspnes J., Wait-free Consensus with Infinite Arrivals. *Proc. 34th Symposium on Theory of Computing (STOC'02)*, ACM Press, pp. 524-533, 2002.

[6] Attiya H., Gorbach A. and Moran S., Computing in Totally Anonymous Asynchronous Shared Memory Systems. *Information and Computation*, 173(2):162-183, 2002.

[7] Attiya H., Snir M. and Warmuth M.K., Computing on an Anonymous Ring. *Journal of the ACM*, 35(4):845-875, 1988.

[8] Attiya H. and Welch J., Distributed Computing, Fundamentals, Simulation and Advanced Topics (Second edition). *Wiley Series on Parallel and Distributed Computing*, 414 pages, 2004.

[9] Borowsky E. and Gafni E., Generalized FLP Impossibility Results for $t$-Resilient Asynchronous Computations. *Proc. 25th ACM Symposium on Theory of Computation (STOC'93)*, California (USA), pp. 91-100, 1993.

[10] Buhrman H., Panconesi A., Silvestri R. and Vityani P., On the Importance of Having an Identity or Is Consensus Really Universal? *Distributed Computing*, 18(3):167-175, 2006.

[11] Chandra T. and Toueg S., Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM*, 43(2):225-267, 1996.

[12] Chandra T., Hadzilacos V. and Toueg S., The Weakest Failure Detector for Solving Consensus. *Journal of the ACM*, 43(4):685–722, 1996.

[13] Chaudhuri S., More *Choices* Allow More *Faults:* Set Consensus Problems in Totally Asynchronous Systems. *Information and Computation*, 105:132-158, 1993.

[14] Chaudhuri S., Herlihy M., Lynch N. and Tuttle M., Tight Bounds for $k$-Set Agreement. *Journal of the ACM*, 47(5):912-943, 2000.

[15] Chothia T. and Chatzikokolakis K., A Survey of Anonymous Peer-to-Peer File-Sharing. *Proc. Satellite workshop of the Int'l Conference on Embedded and Ubiquitous Systems (EUS'05)*, pp. 744-755, 2005.

[16] Delporte-Gallet C., Fauconnier H. and Guerraoui R., A Realistic Look at Failure Detectors. *Proc. IEEE Int'l Conference on Dependable Systems and Networks (DSN'02)*, IEEE Computer Society Press, pp. 345-352, 2002.

[17] Dolev D., Reischuk R. and Strong R., Early Stopping in Byzantine Agreement. *Journal of the ACM*, 37(4):720-741, April 1990.

[18] Durresi A., Paruchuri V., Durresi M. and Barolli L., A Hierarchical Anonymous Communication Protocol for Sensor Networks. *Proc. Int'l Conference on Embedded and Ubiquitous Systems (EUS'05)*, Springer verlag LNCS #3824, pp. 1123-1132, 2005.

[19] Dwork C. and Moses Y., Knowledge and Common Knowledge in a Byzantine Environment: Crash Failures. *Information and Computation*, 88(2):156-186, 1990.

[20] Elrad T.E. and Francez N., Decomposition of Distributed Programs into Communication-Closed Layers. *Science of Computer Programming*, 2(3):155-173, 1982.

[21] Fischer M.J., Lynch N.A., A Lower Bound on the Time to Assure Interactive Consistency. *Information Processing Letters*, 14(4):183-186, 1982.

[22] Fischer M.J., Lynch N.A. and Paterson M.S., Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM*, 32(2):374-382, 1985.

[23] Gafni E., Round-by-round Fault Detectors: Unifying Synchrony and Asynchrony. *Proc. 17th ACM Symposium on Principles of Distributed Computing (PODC'00)*, ACM Press, pp. 143-152, 1998.

[24] Guerraoui R. and Raynal M., The Alpha of indulgent consensus. *The Computer Journal*, 50(1):53-67, 2007.

[25] Guerraoui R. and Ruppert E., Anonymous and Fault-tolerant Shared Memory Computing. *Distributed Computing*, 20(3):165-177, 2007.

[26] Halpern J.Y. and Moses Y., Knowledge and Common Knowledge in a Distributed Environment. *Journal of the ACM*, 37(3):549-587, 1990.

[27] Hélary J.-M., Hurfin M., Mostefaoui A., Raynal M. and Tronel F. Computing Global Functions in Asynchronous Distributed Systems with Perfect Failure Detectors. *IEEE Transactions on Parallel and Distributed Systems*, 11(9):897-909, 2000.

[28] Herlihy M.P., Wait-Free Synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):124-149, 1991.

[29] Herlihy M.P., Luchangco V. and Moir M., Obstruction-Free Synchronization: Double-ended Queues as an Example. *Proc. 23th Int'l IEEE Conference on Distributed Computing Systems (ICDCS'03)*, pp. 522-529, 2003.

[30] Herlihy M.P. and Shavit N., The Topological Structure of Asynchronous Computability. *Journal of the ACM*, 46(6):858-923, 1999.

[31] Keidar I. and Rajsbaum S., A Simple Proof of the Uniform Consensus Synchronous Lower Bound. *Information Processing Letters*, 85:47-52, 2003.

[32] Lakshman T.V. and Wei V.K., Distributed Computing on Regular Networks with Anonymous Nodes. *IEEE Transactions on Computers*, 43(2):211-218, 1994.

[33] Loui M.C., Abu-Amara H., Memory Requirements for Agreement Among Unreliable Asynchronous Processes. *Advances in Computing research*, JAI Press, 4:163-183, 1987.

[34] Lynch N.A., Distributed Algorithms. *Morgan Kaufmann Pub.*, San Francisco (CA), 872 pages, 1996.

[35] Moses Y. and Raynal M., No Double Discount: Condition-Based Simultaneity Yields Limited Gain. *Proc. 22th Int'l Symposium on Distributed Computing (DISC'08)*, Springer-Verlag LNCS #5218, pp. 423-437, 2008.

[36] Moses Y. and Tuttle M.R., Programming Simultaneous Actions Using Common Knowledge. *Algorithmica*, 3:121-169, 1988.

[37] Mostefaoui A., Rajsbaum S., Raynal M. and Travers C., On the Computability Power and the Robustness of Set Agreement-oriented Failure Detector Classes. *Distributed Computing*, 21(3):201-222, 2008.

[38] Mostefaoui A., Rajsbaum S., Raynal M. and Travers C., The Combined Power of Conditions and Information on Failures to Solve Asynchronous Set Agreement. To appear in *SIAM Journal of Computing*, 2009.

[39] Mostefaoui A. and Raynal M., Leader-Based Consensus. *Parallel Processing Letters*, 11(1):95-107, 2001.

[40] Panconesi A., Papatriantafilou M., Tsigas Ph. and Vityani P., Randomized Naming Using Wait-free Shared Variables. *Distributed Computing*, 11(3):113-124, 1998.

[41] Raynal M., Consensus in Synchronous Systems: a Concise Guided Tour. *Proc. 9th IEEE Pacific Rim Int'l Symposium on Dependable Computing (PRDC'02)*, IEEE Computer Press, pp. 221-228, 2002.

[42] Saks M. and Zaharoglou F., Wait-Free $k$-Set Agreement is Impossible: The Topology of Public Knowledge. *SIAM Journal on Computing*, 29(5):1449-1483, 2000.

[43] Yamashita M. and Kameda T., Computing on Anonymous Networks: Part I -Characterizing the Solvable Cases. *IEEE Transactions on Parallel Distributed Systems*, 7(1):69-89, 1996.

[44] Yamashita M. and Kameda T., Computing on Anonymous Networks: Part II -Decision and Membership Problems. *IEEE Transactions on Parallel Distributed Systems*, 7(1):90-96, 1996.

# A On the formal definition of $\psi$

This section presents a formal definition of the failure detector class $\psi$ that fits into the formal failure detector framework defined in [11].

**On the framework** The definitions that follows assume the following framework.

- As in [11], the framework provides us with a discrete global clock whose domain is the set $N$ of positive integers. (The notation $\tau$ is used to denote a time value.) As indicated in [11], this time notion can be used to reason on the behavior of failure detector-based algorithms, but remains always unknown by the processes.
- The framework provides us with a set of indexes $\mathcal{I} = \{1, \ldots, n\}$ such that each process has a unique index in $\mathcal{I}$ (an index can be interpreted as a process identity/name). This allows us to use the notation $p_i$ to denote a given process and distinguish it from another process $p_j$. In [11], these indexes are known by the processes that can use them (e.g., to select a rotating coordinator). Differently, in the anonymous model defined here, the processes do not known the existence of the indexes, and consequently -similarly to the global clock- these indexes cannot be used in the algorithms. This constitutes a fundamental difference between anonymous and non-anonymous systems.

**Chandra and Toueg's definitions** The following definitions are from [11].

- A *failure pattern* is a function $F : N \to 2^{\mathcal{I}}$ where $F(\tau)$ denotes the set of processes that have crashed through time $\tau$. As no crashed process recovers, we have $F(\tau) \subseteq F(\tau + 1)$.

  Given a run, let $Faulty = \cup_{\tau \geq 0} F(\tau)$ (the indexes of the processes that crash during that run), and $Correct = \mathcal{I} \setminus Faulty$ (the indexes of the processes that do not crash during that run).
- A *failure detector history with range* $R$ describes the behavior of a failure detector during a run. It is a function $H : \mathcal{I} \times N \to R$ where $H(i, \tau)$ describes the value of the failure detector at $p_i$ at time $\tau$. That value belongs to $R$.
- A *failure detector* $\mathcal{D}$ *with range* $R$ is a function that maps each failure pattern $F$ to a set of failure detector histories with range $R$: $\mathcal{D}(F)$ is the set of failure detector histories that $\mathcal{D}$ can exhibit when the failure pattern is $F$.

**The failure detector class** $\psi$ The range $R$ of the class $\psi$ is the set of integers $\{1, \ldots, n\}$. For every failure pattern $F$ we have:

- Safety. $\forall \tau : \forall i \notin F(\tau) : H(i, \tau) \geq n - |F(\tau)|$.
- Liveness. $\exists \tau : \forall \tau' \geq \tau : \forall i \notin F(\tau') : H(i, \tau') = n - |Faulty|$.

# B On the hierarchy of failure detectors to solve consensus despite anonymity

**A hierarchy of failure detector classes in non-anonymous systems** In non-anonymous asynchronous systems, the failure detector classes denoted $\mathcal{P}$, $\diamond\mathcal{P}$ and $\Omega$ (defined in [11, 12]) define a strict hierarchy, namely we have $\mathcal{P} \subset \diamond\mathcal{P} \subset \Omega$. These three failure detector classes output process names. $\mathcal{P}$ is the class of perfect failure detectors (they never suspect a process while it is alive, and eventually suspect all the crashed processes). $\diamond\mathcal{P}$ is the class of eventually perfect failure detectors (they is a finite time after which they behave as a perfect failure detector). $\Omega$ is the class of eventual leader failure detectors. It includes all the failure detectors that provide each process $p_i$ with a local variable $leader_i$, that $p_i$ can only read. At any time, each local variable $leader_i$ contains a process name, and there is a finite time after which the local variables $leaders_i$ of all the correct processes contain forever the same process name that is the name of a correct process (before that time they can behave arbitrarily). The fact that $\mathcal{P} \subset \diamond\mathcal{P}$ follows from their definition. the fact that $\diamond\mathcal{P} \subset \Omega$ is a consequence of transformations given in several papers (e.g., [12]).

Moreover, as far as the consensus problem is concerned, it is shown in [12] that $\Omega$ is the weakest class of failure detectors that allows solving consensus despite asynchrony and process crashes in non-anonymous systems where $t < n/2$.

Irisa

**Is there a hierarchy in anonymous systems?** As indicated in the introduction of the paper, if we give distinct names to the processes of an initially anonymous system, the failure detector classes $\mathcal{P}$ and $\psi$ are equivalent [37, 38], which means that, in a non-anonymous asynchronous system, given a failure detector of any of the classes $\mathcal{P}$ and $\psi$, it is possible to construct a failure detector of the other class.

It is easy to define a class of anonymous eventual leader failure detectors that is the counterpart of $\Omega$ for anonymous systems. Let $\Omega_A$ denote that class. It is defined as follows. Each process $p_i$ is provided with a boolean $leader_i$ (that $p_i$ can only read) such that, after some finite time, the boolean of a correct process remains forever true (this process is not known in advance and can never be explicitly known) while the boolean of all the other processes remain forever false (during an arbitrary long period of time, the values of the boolean local variables can be arbitrary). Such an anonymous "view" of the $\Omega$ class is used in several papers (e.g., [24]). It appears that a slight modification of the $\Omega$-based consensus algorithm described in [39] works for anonymous systems where $n$ and $t$ are known, and $\Omega$ is replaced by $\Omega_A$.

The previous observations set two questions that state open problems (for future research).

- As (1) $\psi$ and $\Omega_A$ are the anonymous counterparts of $\mathcal{P}$ and $\Omega$, respectively, and (2) $\Omega$ is weaker than $\mathcal{P}$, a natural question is the following: Is there a relation linking $\psi$ and $\Omega_A$? More precisely, is $\psi$ stronger than $\Omega_A$ (in an anonymous system)? Up to now, we do not know if $\psi$ and $\Omega_A$ can be compared.
- Another important question is related to the lower bound on information on failures, more precisely, which is the (or is there a) weakest failure detector class for solving the consensus problem in an anonymous system?

# C An impossibility in the model $\mathcal{AARS}^{cl}_{n,t}[\psi]$

This appendix shows that, in the round communication-closed model $\mathcal{AARS}^{cl}_{n,t}[\psi]$, there is no early deciding consensus algorithm in which the processes decide and halt in $af + b$ rounds, where $a$ and $b$ are any predefined constant values.

**Lemma 16** *Let $1 \leq t < n - 1$. There is no algorithm that allows the processes to decide and halt in $t + 1$ rounds when $f = 0$ (and $2t + 1$ rounds otherwise) in the model $\mathcal{AARS}^{cl}_{n,t}[\psi]$.*

**Proof** The proof is by contradiction. Let us suppose that there is an algorithm $A$ that allows the processes to decide and halt in $t + 1$ rounds when there is no crash ($f = 0$). Starting from a failure-free synchronous run $R_0$, we build successive runs $R_1, \ldots, R_t$ in which the processes must decide and halt as in $R_0$. The contradiction is obtained in the last run $R_t$ where processes must decide a value they have never received.

**Run $R_0$** Let us consider the runs of $A$ where each process $p_i$ proposes the value $i$ ($1 \leq i \leq n$). Let $R_0$ be a crash-free run of $A$ whose behavior is exactly the same as the one in a pure synchronous system. This run is represented on Figure 9 (where the value proposed by each process appears at the left of its time line, and the integer -here $n$- at the end of a round indicates how many messages have been received by the corresponding processes by the end of that round). Let us suppose that, without loss of generality, all the processes decide the value 1 after $t + 1$ rounds (indicated by $D(1)$ in the figure).

**From $R_0$ to $R_1$** Let $R'_0$ be a run identical to $R_0$ up to round $t$, and where during the round $t + 1$, the $t$ processes $p_1, \ldots, p_t$ crash after having sent their round $t + 1$ message (Figure 10(a)). Moreover, in $R'_0$, the processes $p_{t+1}, \ldots p_{n-1}$ ([8]) receive $n$ round $t + 1$ messages. As they cannot distinguish $R'_0$ from $R_0$, they decide the same value (i.e., 1) in both runs. In $R'_0$, differently from the other processes, the process $p_n$

[8] Actually at least one process in this set is sufficient.

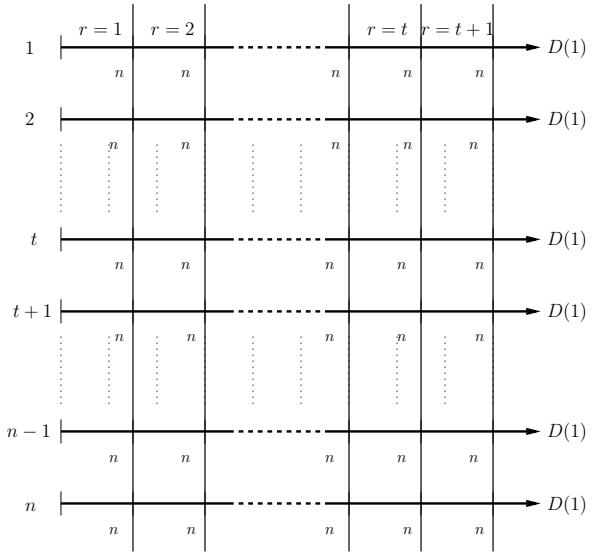
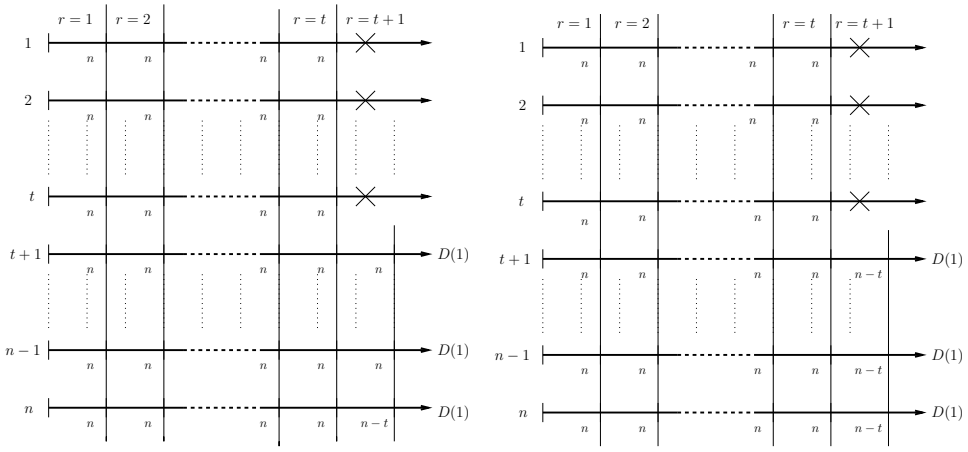



Figure 9: Run $R_0$

receives only $n-t$ round $t+1$ messages (let us notice that these $n-t$ messages may come from any subset of processes that start round $t+1$, including the processes that crash during that round). As indicated in Figure 10(a), the process $p_n$ decides 1 as the other correct processes.

Let $R_0''$ (Figure 10(b)) a run identical to $R_0'$ except that all the processes $p_{t+1}, \ldots p_n$ behave as $p_n$, i.e., each of them receives $n-t$ round $t+1$ messages from any subset of processes that start round $t+1$. As $p_n$ cannot distinguish $R_0'$ and $R_0''$, it follows that the correct processes $p_{t+1}, \ldots p_n$ decide 1 in $R_0''$.



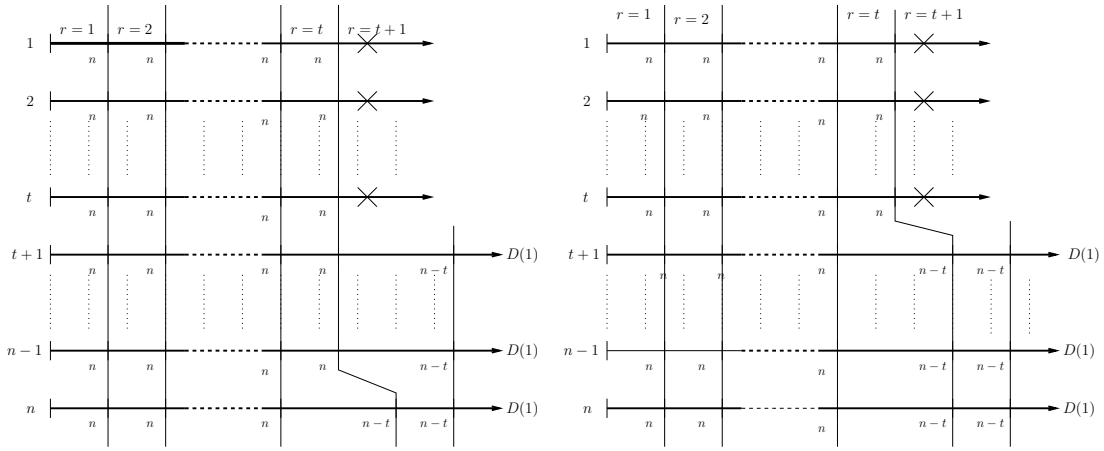

(a) Run $R_0'$ (b) Run $R_0''$

Figure 10: From the run $R_0$ to the run $R_0''$

Let us now consider the run $R_0'''$ (Figure 11(a)) that is identical to $R_0''$ up to round $t-1$. Then, the $t$ processes $p_1, \ldots, p_t$ receives $n$ round $t$ messages, proceed to the round $t+1$ and crash. The processes $p_{t+1}, \ldots, p_{n-1}$ receive $n$ round $t$ messages (in round $t$), and any subset of $n-t$ round $t+1$ messages (in round $t+1$) from the processes $p_1, \ldots, p_{n-1}$ (all but $p_n$). Differently, the correct process $p_n$ is informed

(due to its failure detector) of the $t$ crashes (of the processes $p_1, \ldots, p_t$) while it is still in round $t$, and consequently receives only $n-t$ round $t$ messages (in round $t$) before proceeding to the round $t+1$ , during which it receives the same $n-t$ round $t+1$ messages as in $R_0''$. Since no correct process (but $p_n$) can distinguish $R_0''$ and $R_0'''$ and as there are at least two correct processes (because $t < n-1$), all the correct processes have to decide 1. Hence, $p_n$ decides 1.

Let us now define the run $R_1$ that is identical to $R_0''$ up to round $t-1$, and where during the rounds $t$ and $t+1$, the processes $p_{t+1}, \ldots p_n$ behave as $p_n$ behaves in $R_0'''$ (each of them receives a set of $n-t$ round $t$ messages during the round $r$ and a set of $n-t$ round $t+1$ messages during the round $t+1$. It follows from the previous discussion that all the correct processes decide 1 in $R_1$.




(a) Run $R_0'''$ (b) Run $R_1$

Figure 11: From the run $R_0$ to the run $R_1$

**From the run $R_{k-1}$ to the run $R_k$** We now define the run $R_k$ from the run $R_{k-1}$ for $1 \leq k \leq t$. $R_k$ is defined as follows.

- $\forall\, x$ such that $t-k+1 \leq x \leq t$, the process $p_x$ crashes in the round $x+1$, and $\forall\, x$ such that $1 \leq x \leq t-k$, the process $p_x$ crashes in round $t-k+2$. The other processes do not crash.

- All the processes terminate the rounds from 1 to $t-k$ by receiving $n$ messages in each of these rounds. In others words, up to the round $t-k$, $R_k$ is identical to $R_0$ (which is crash-free).

- During the round $t-k+1$, the processes $p_1, \ldots, p_{t-k+1}$ receive $n$ messages and then crash while they are in the round $t-k+2$. Moreover, during the round $t-k+1$, each of the remaining processes (i.e., each of $p_{t-k+2}, \ldots, p_n$) receives $n-(t-k+1)$ round $t-k+1$ messages.

- In any round $r$ such that $t-k+2 \leq r \leq t$, the process $p_r$ (1) receives $n-r+1$ round $r$ messages; (2) enters the round $r+1$ (let us observe that this is possible because, at that time, there are exactly $n-r+1$ non-crashed processes); (3) and then crashes while executing the round $r+1$.

  Moreover, during that round $r$, each of the remaining processes (i.e., each of $p_{r+1}, \ldots, p_n$) is informed of that crash by its failure detector, and enters the round $r+1$ by receiving only $n-r$ round $r$ messages.

- In the last round $t+1$, the process $p_t$ crashes and this crash allows the processes $p_{t+1}, \ldots, p_n$ to receive only $n-t$ round $t$ messages in order to enter the round $t+1$. Each of these $n-t$) processes receives again $n-t$ round $t+1$ messages, and decides.

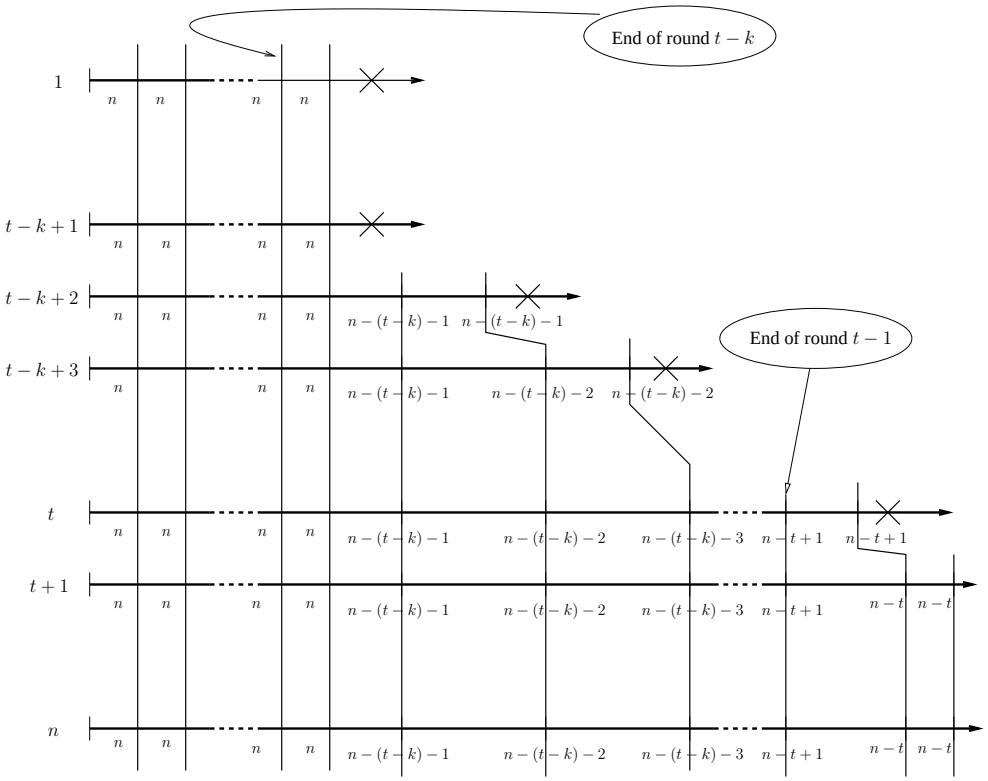



Figure 12: Run $R_k$

The same pattern as the one employed to go from $R_0$ to $R_1$, allows to go from $R_{k-1}$ to $R_k$, for $1 \leq k \leq t$. Moreover, for any such $k$, all the correct decide at the end of the round $t+1$ of $R_k$ the same value as the one they decide in $R_{k-1}$.

**Obtaining the contradiction** In the run $R_t$, the correct processes must decide as in $R_0$, i.e., they must decide value 1. However, there exists a particular run $R'_t$, where the set of messages received by all processes does not include the value proposed by $p_1$. More precisely, in the first round of $R'_t$, all the processes but $p_1$ receive $n-1$ messages (one from each other but $p_1$). In the second round of $R'_t$, all the processes but $p_1$ receive $n-1$ messages (one from each other but $p_1$), and $p_1$ crashes during that round. It is clear that no process will ever know the value proposed by $p_1$, and consequently no correct process can decide the value 1 proposed by $p_1$ (recall that only $p_i$ proposes the value $i$). As, due its very construction, $R'_t$ is $R_t$, the contradiction follows. $\square_{Lemma\ 16}$

**Theorem 8** *There is no algorithm in the model $\mathcal{AARS}^{cl}_{n,t}[\psi]$ where the processes decide and halt in at most $af+b$ rounds where $a$ and $b$ are predefined constant values.*

**Proof** The proof follows from Lemma 16 that has shown that $t+1$ rounds are necessary when $f=0$. $\square_{Theorem\ 8}$